hw2

Jiajian Huang

2/25/2019

HW4

Q9

a

```
set.seed(2019)
n=nrow(College)
x=c()

train=sample(1:n,n/2)
test=-train
```

b

```
lm.q9fit=lm(Apps~.,data=College[train,])
lm.q9pred=predict(lm.q9fit,newdata = College[test,])
x=c(x,lm=mean( (lm.q9pred-College[test,"Apps"])^2 ))
```

c

```
grid = 10 ^ seq(4, -2, length=100)
train.mat = model.matrix(Apps~., data=College[train,])
test.mat = model.matrix(Apps~., data=College[test,])
```

```
ridge.cv=cv.glmnet(x=train.mat,y=College[train,"Apps"],alpha=0,thresh=1e-12,lambda =
grid)
ridge.fit=glmnet(x=train.mat,y=College[train,"Apps"],alpha=0,lambda = ridge.cv$lambda
.min)
ridge.pred=predict(ridge.fit,newx=test.mat )
ridge.coef=predict(ridge.fit,type='coefficients',s=ridge.cv$lambda.min)
```

```
ridge.cv$lambda.min
```

```
## [1] 0.01
```

```
ridge.coef
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  -32.95543909
## (Intercept)    .
## PrivateYes  -615.40959574
## Accept         1.79867709
## Enroll        -1.17861001
## Top10perc     57.70411961
## Top25perc    -13.96484959
## F.Undergrad    0.01769362
## P.Undergrad    0.05055899
## Outstate      -0.08895161
## Room.Board     0.18695405
## Books         -0.14810737
## Personal       0.03630867
## PhD          -11.76973565
## Terminal      -4.47815269
## S.F.Ratio      7.12455039
## perc.alumni   -3.21716861
## Expend         0.07406544
## Grad.Rate      7.60971314
```

```
x=c(x,ridge=mean( (ridge.pred-College[test,2])^2))
```

d

```
lasso.cv=cv.glmnet(train.mat,y=College[train,"Apps"],alpha=1,lambda=grid,thresh=1e-12
)
lasso.fit=glmnet(x=as.matrix(College[train,-c(1,2)]),y=College[train,2],alpha=1,lambd
a = lasso.cv$lambda.min)
lasso.coef=predict(lasso.fit,type='coefficients',s=lasso.cv$lambda.min)

lasso.cv$lambda.min
```

```
## [1] 0.01
```

```
lasso.coef
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -632.13627083
## Accept         1.78974219
## Enroll        -1.15721457
## Top10perc     56.92579533
## Top25perc    -13.23164850
## F.Undergrad    0.04218071
## P.Undergrad    0.05758272
## Outstate      -0.11454170
## Room.Board     0.15820896
## Books         -0.16621620
## Personal       0.04990224
## PhD          -10.30498306
## Terminal      -1.05571513
## S.F.Ratio     15.98407642
## perc.alumni   -4.37534863
## Expend         0.07734259
## Grad.Rate      6.33799024
```

```
lasso.pred=predict(ridge.fit,newx=test.mat)
x=c(x,lasso=mean( (lasso.pred-College[test,2])^2))
```

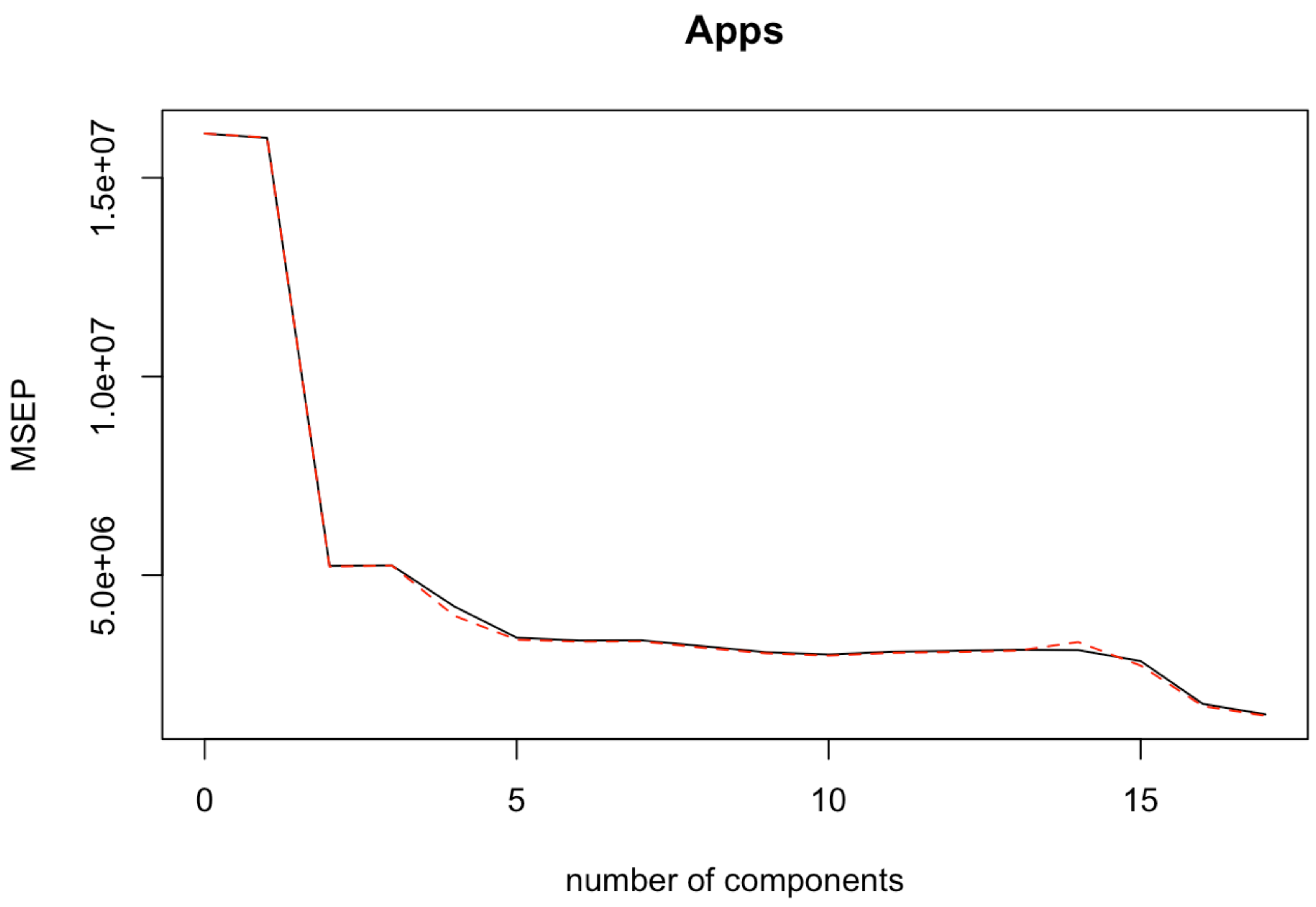
e

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

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

```
set.seed(2019)

pcr.fit=pcr(Apps~.,data=College,scale=TRUE,validation='CV',subset=train)
validationplot(pcr.fit,val.type="MSEP")
```



```
pcr.fit$ncomp
```

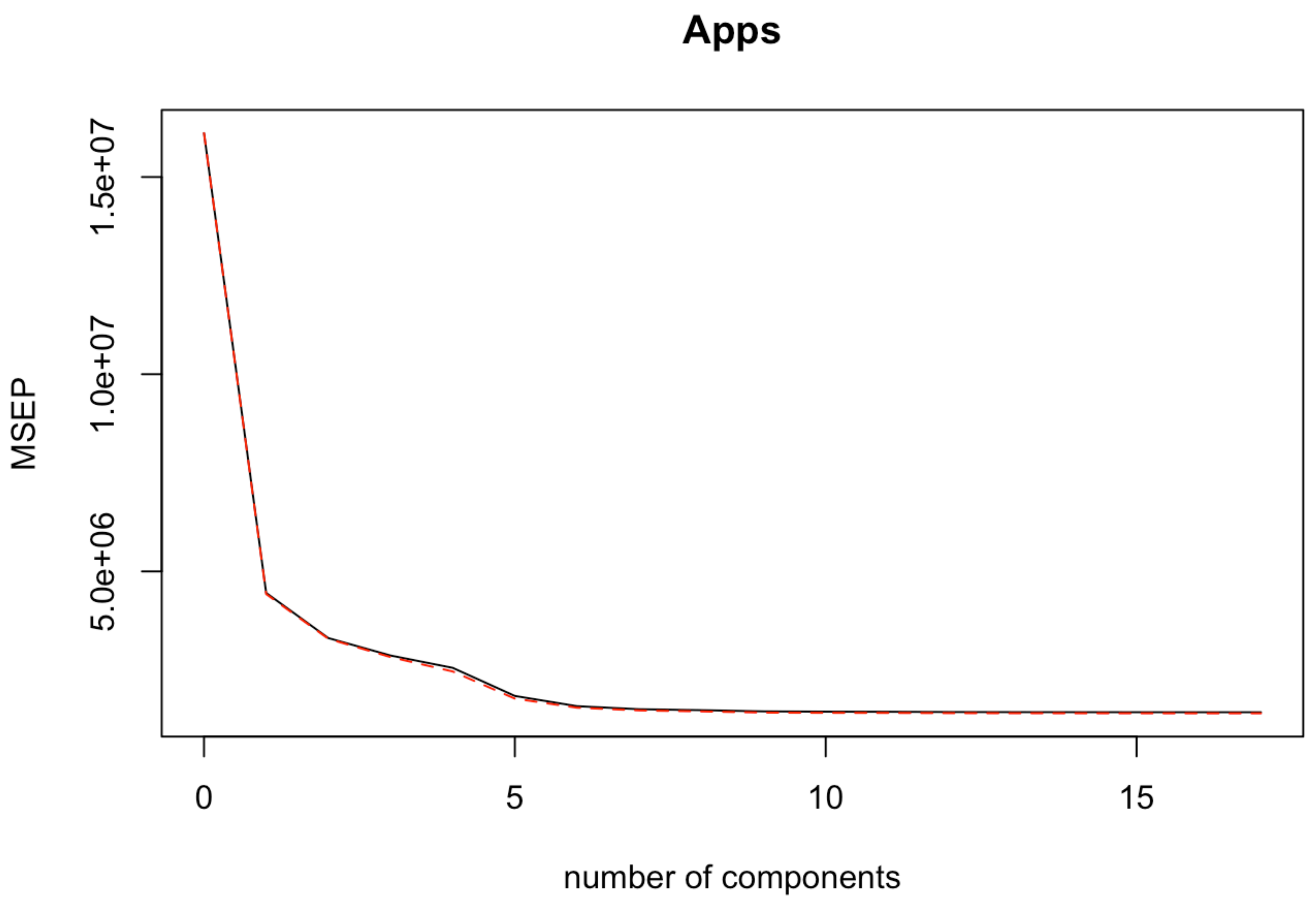
```
## [1] 17
```

From the plot and the object returned we know that the number of components that achieved the lowest cross validation error is 17. The test error is given below.

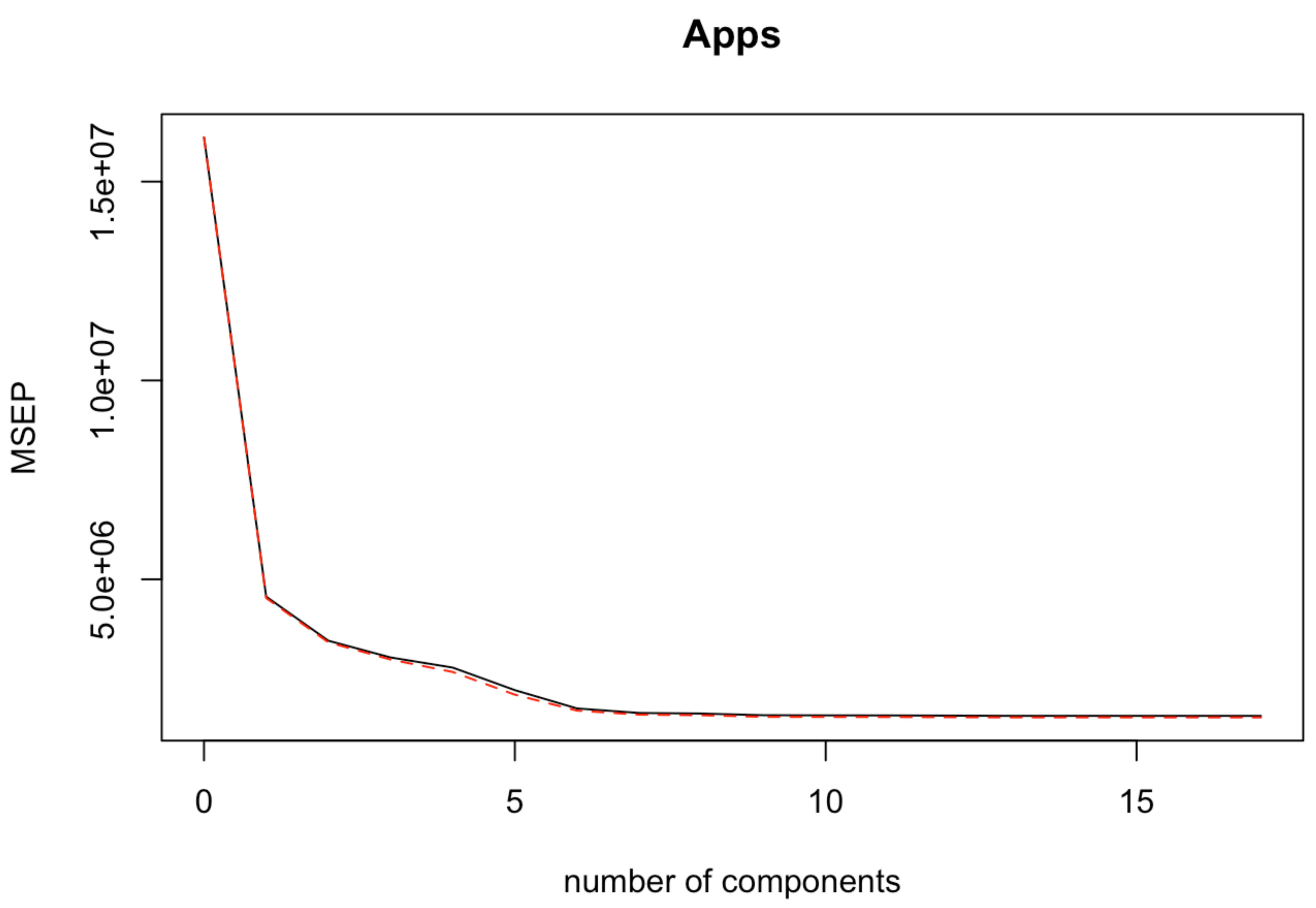
```
pcr.pred=predict(pcr.fit,newdata = College[test,-c(2)],ncomp = 16)
x=c(x,pcr=mean( (pcr.pred-College[test,2])^2))
```

f

```
pls.fit=plsr(Apps~.,data=College,scale=TRUE,validation='CV',subset=train)
validationplot(pls.fit,val.type="MSEP")
```



```
pls.fit=plsr(Apps~.,data=College,scale=TRUE,validation='CV',subset=train)
validationplot(pls.fit,val.type="MSEP")
```



```
pls.pred=predict(pls.fit,newdata = College[test,-c(2)],ncomp = 10)
x=c(x,pls=mean( (pls.pred-College[test,2])^2))
```

g

```
sort(x)
```

```
##   ridge   lasso      lm     pls     pcr
## 1248924 1248924 1250572 1260706 1298244
```

From the results obtained there is not a significant difference from fitting a model with least squares, ridge, lasso and partial least squares. The lasso and ridge regression significantly penalize the Books, Personal, Terminal and S.F. Ratio predictors. We can see that these are also not found to be significant in the least squares model.

```
summary(lm.q9fit)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = College[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3740.0  -435.2   -20.7   326.1  7187.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -27.53333  575.21654  -0.048 0.961849
## PrivateYes  -616.52781  200.81813  -3.070 0.002298 **
## Accept         1.80111    0.05225  34.473  < 2e-16 ***
## Enroll        -1.19235    0.24888  -4.791 2.41e-06 ***
## Top10perc     57.84937    8.43084   6.862 2.88e-11 ***
## Top25perc    -14.04256    6.26068  -2.243 0.025490 *
## F.Undergrad    0.01912    0.04778   0.400 0.689175
## P.Undergrad    0.05049    0.05952   0.848 0.396886
## Outstate      -0.08893    0.02627  -3.385 0.000787 ***
## Room.Board     0.18663    0.06957   2.683 0.007633 **
## Books         -0.14915    0.33476  -0.446 0.656185
## Personal       0.03665    0.09160   0.400 0.689291
## PhD          -11.80842    6.92624  -1.705 0.089056 .
## Terminal      -4.46100    7.32515  -0.609 0.542900
## S.F.Ratio      7.11515   17.47346   0.407 0.684098
## perc.alumni   -3.18343    5.80967  -0.548 0.584054
## Expend         0.07395    0.01513   4.888 1.52e-06 ***
## Grad.Rate      7.60233    4.19394   1.813 0.070689 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1022 on 370 degrees of freedom
## Multiple R-squared:  0.9378, Adjusted R-squared:  0.935
## F-statistic: 328.4 on 17 and 370 DF,  p-value: < 2.2e-16
```

```
avg_apps=mean(College[,"Apps"])
1 - mean((College[test, "Apps"] - lm.q9pred)^2) /mean((College[test, "Apps"] - avg_ap
ps)^2)
```

```
## [1] 0.9099336
```

The best performing model then errors on average 1248924 and 90% of variance present in the data is explained by the model.

Q11

a

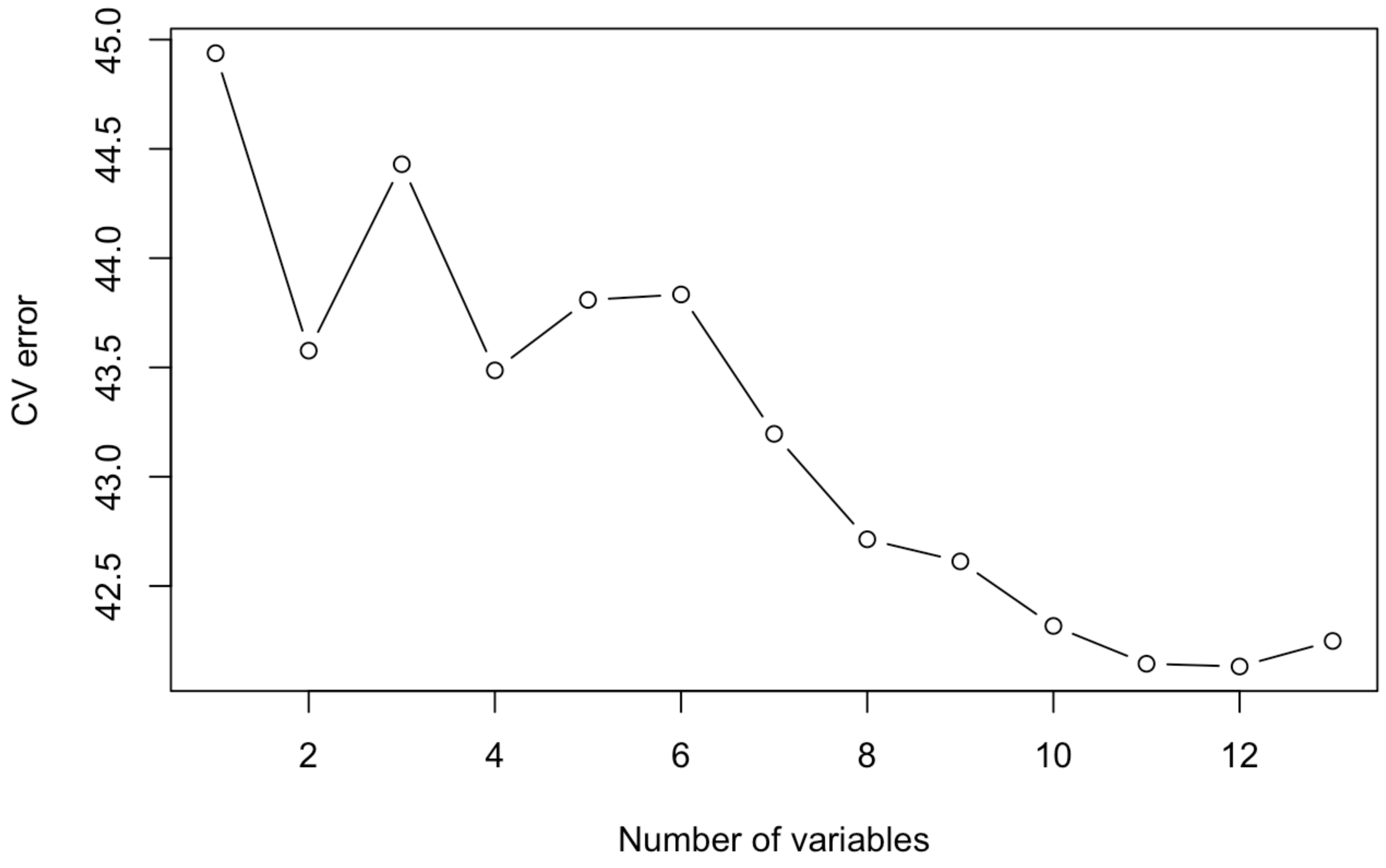
```
library(leaps)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

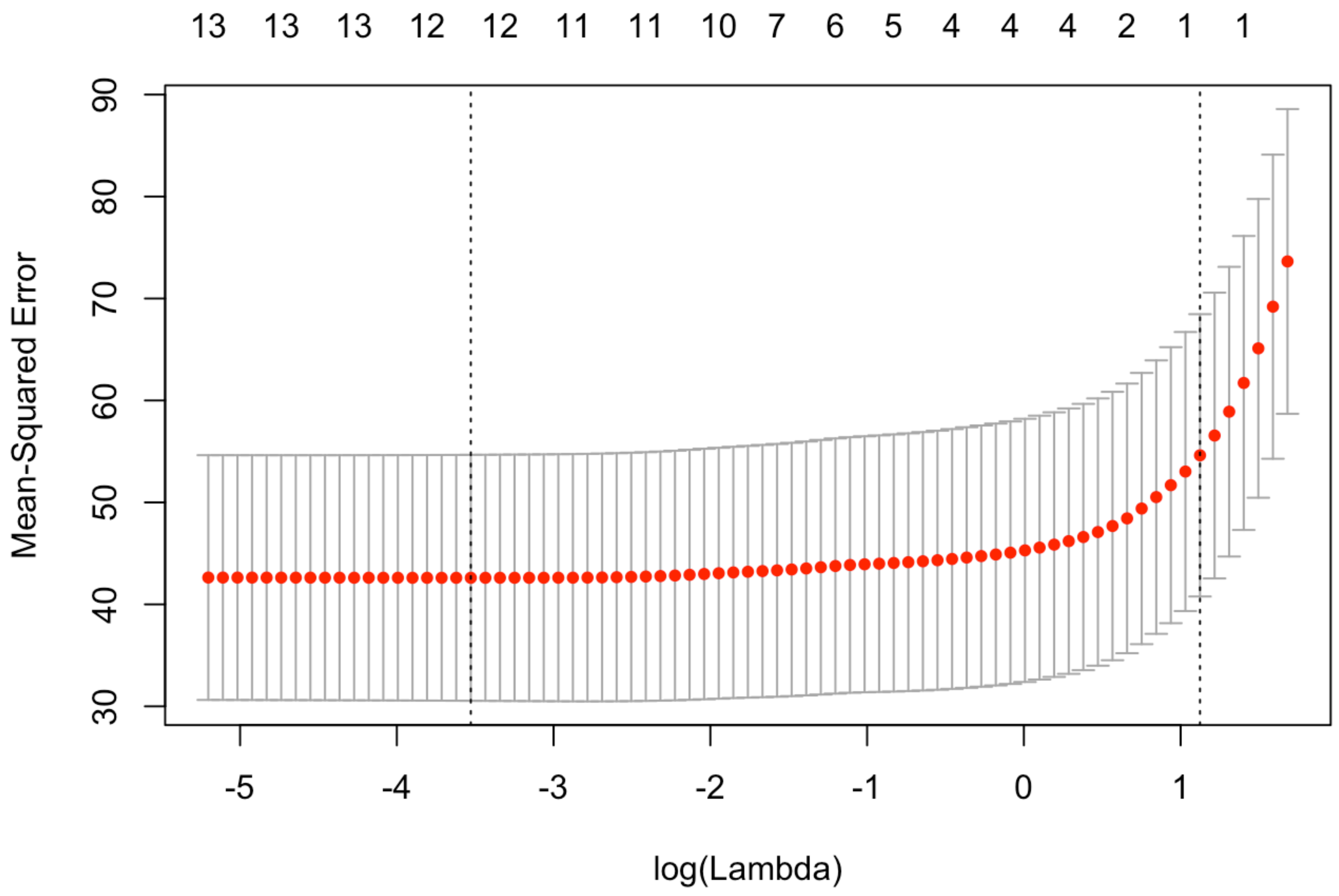
```
library(pls)
library(dplyr)

data("Boston")
predict.regsubsets=function(object, newdata, id, ...) {
    form=as.formula(object$call[[2]])
    mat=model.matrix(form, newdata)
    coefi=coef(object, id = id)
    xvars=names(coefi)
    mat[, xvars] %*% coefi
}
```

```
k = 10
folds=sample(1:k, nrow(Boston), replace = TRUE)
cv.errors=matrix(NA, k, 13, dimnames = list(NULL, paste(1:13)))
for (j in 1:k) {
    best.fit=regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)
    for (i in 1:13) {
        pred=predict(best.fit, Boston[folds == j, ], id = i)
        cv.errors[j, i]=mean((Boston$crim[folds == j] - pred)^2)
    }
}
mean.cv.errors=apply(cv.errors, 2, mean)
plot(mean.cv.errors, type = "b", xlab = "Number of variables", ylab = "CV error")
```



Above the picture, we could see that cross-validation selects an 12-variables model.

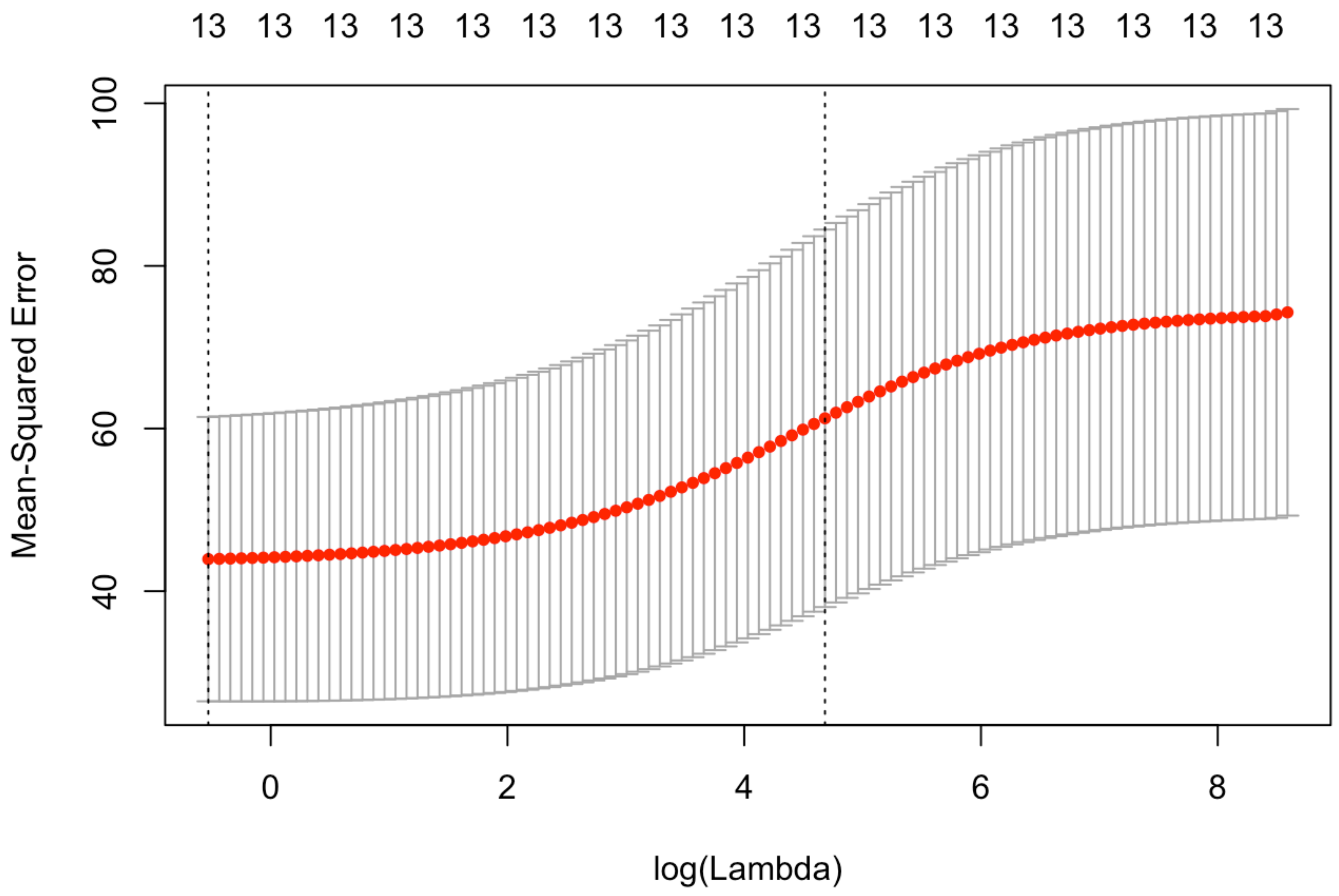
```
x = model.matrix(crim ~ ., Boston)[, -1]
y = Boston$crim
cv.out = cv.glmnet(x, y, alpha = 1, type.measure = "mse")
plot(cv.out)
```



Here cross-validation selects a λ equal to 0.0467489.

Now, we proceed with ridge degression:

```
cv.out <- cv.glmnet(x, y, alpha = 0, type.measure = "mse")
plot(cv.out)
```

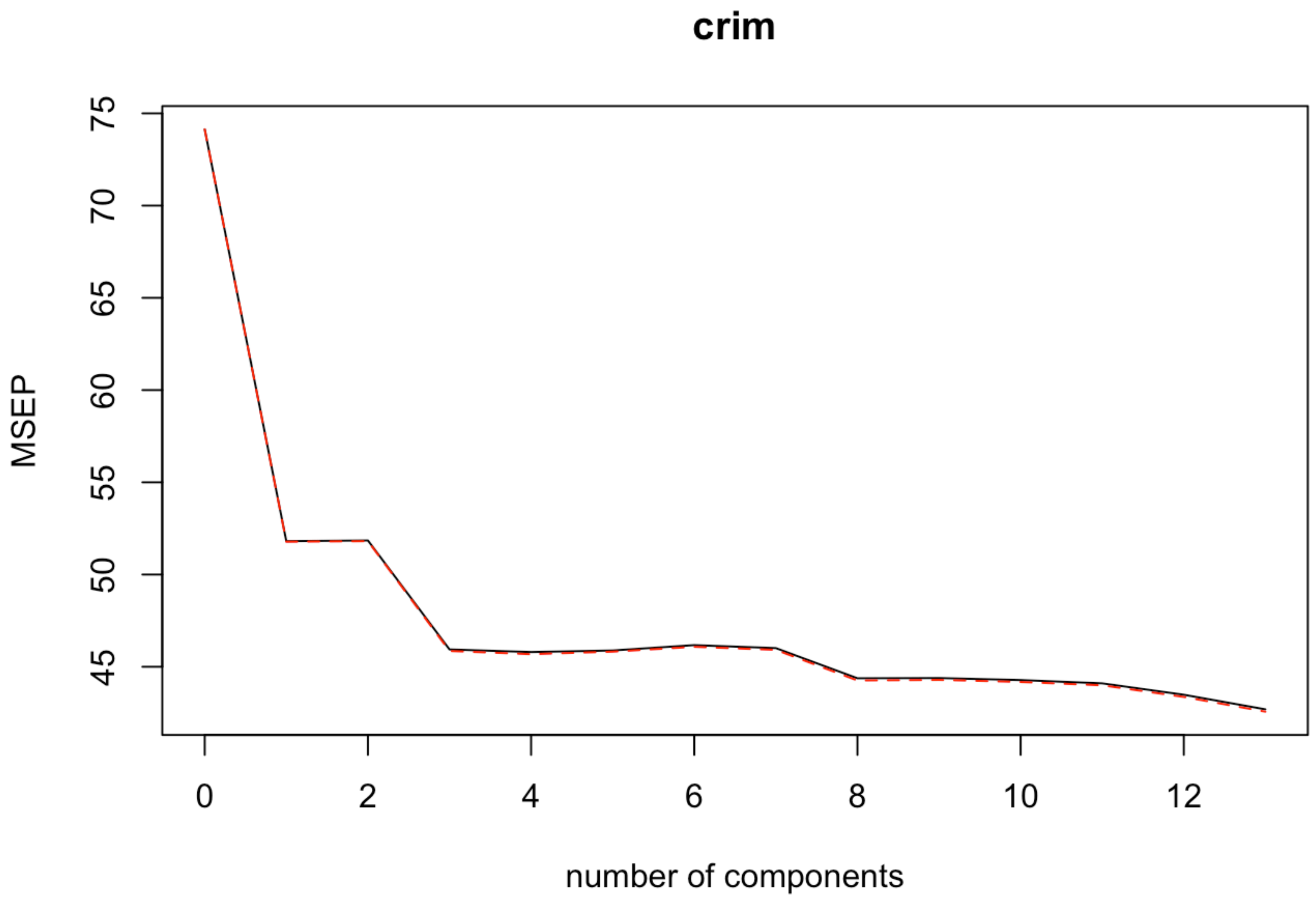


The PCR:

```
pcr.fit <- pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.198    7.201    6.778    6.767    6.774    6.795
## adjCV         8.61    7.196    7.198    6.772    6.759    6.769    6.788
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.783    6.661    6.662     6.654     6.641     6.594     6.534
## adjCV    6.776    6.652    6.655     6.646     6.633     6.585     6.524
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

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



Here cross-validation selects M to be equal to 14.

b

As computed above the model with the lower cross-validation error is the one chosen by the best subset selection method.

c

Not all predictors are strongly related tot the response variable; using all of them will decrease performance since it will overfit the model.

HW 5 Chp 7

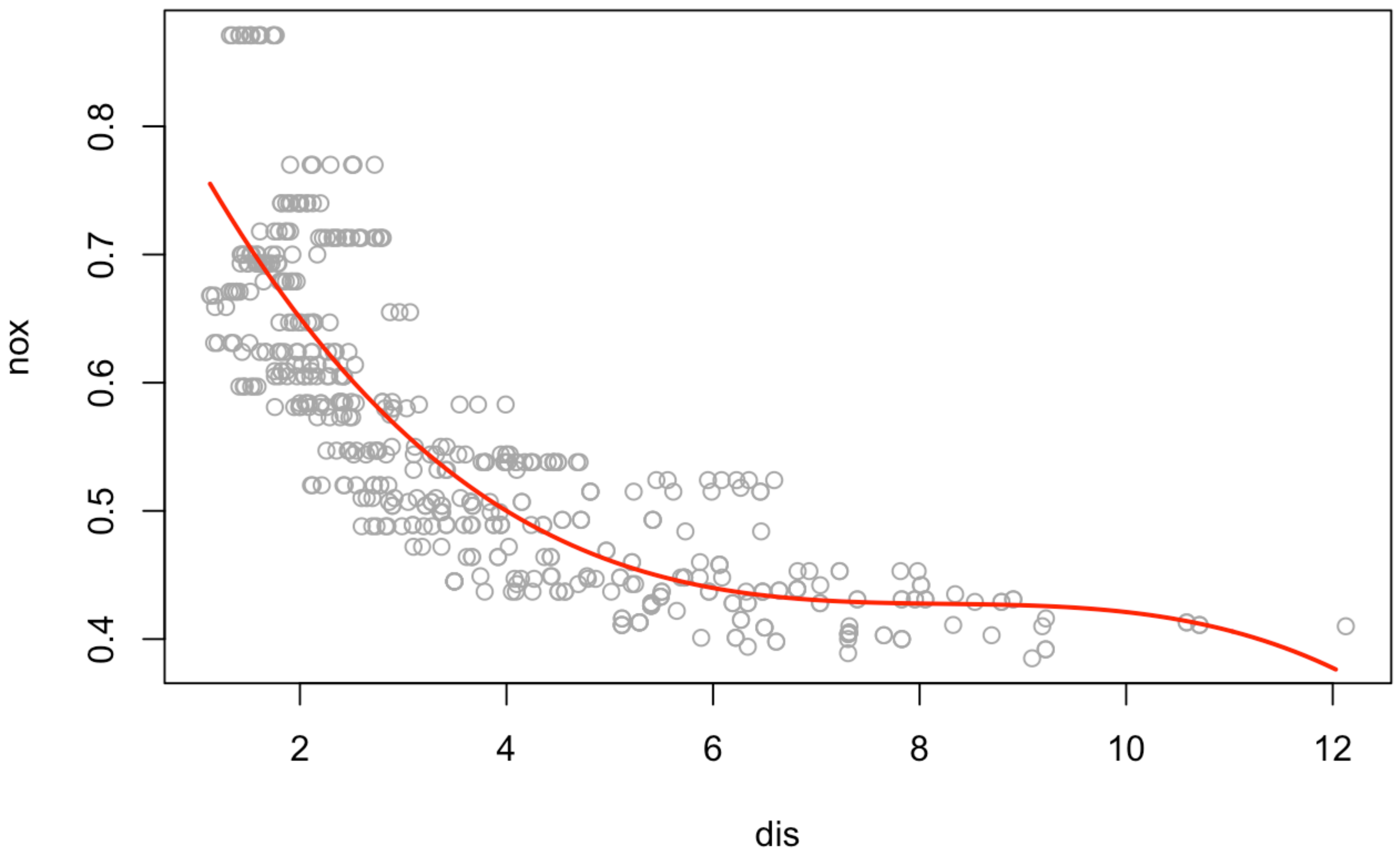
Q9

a

```
set.seed(2020)
fit=lm(nox ~ poly(dis, 3), data = Boston)
summary(fit)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

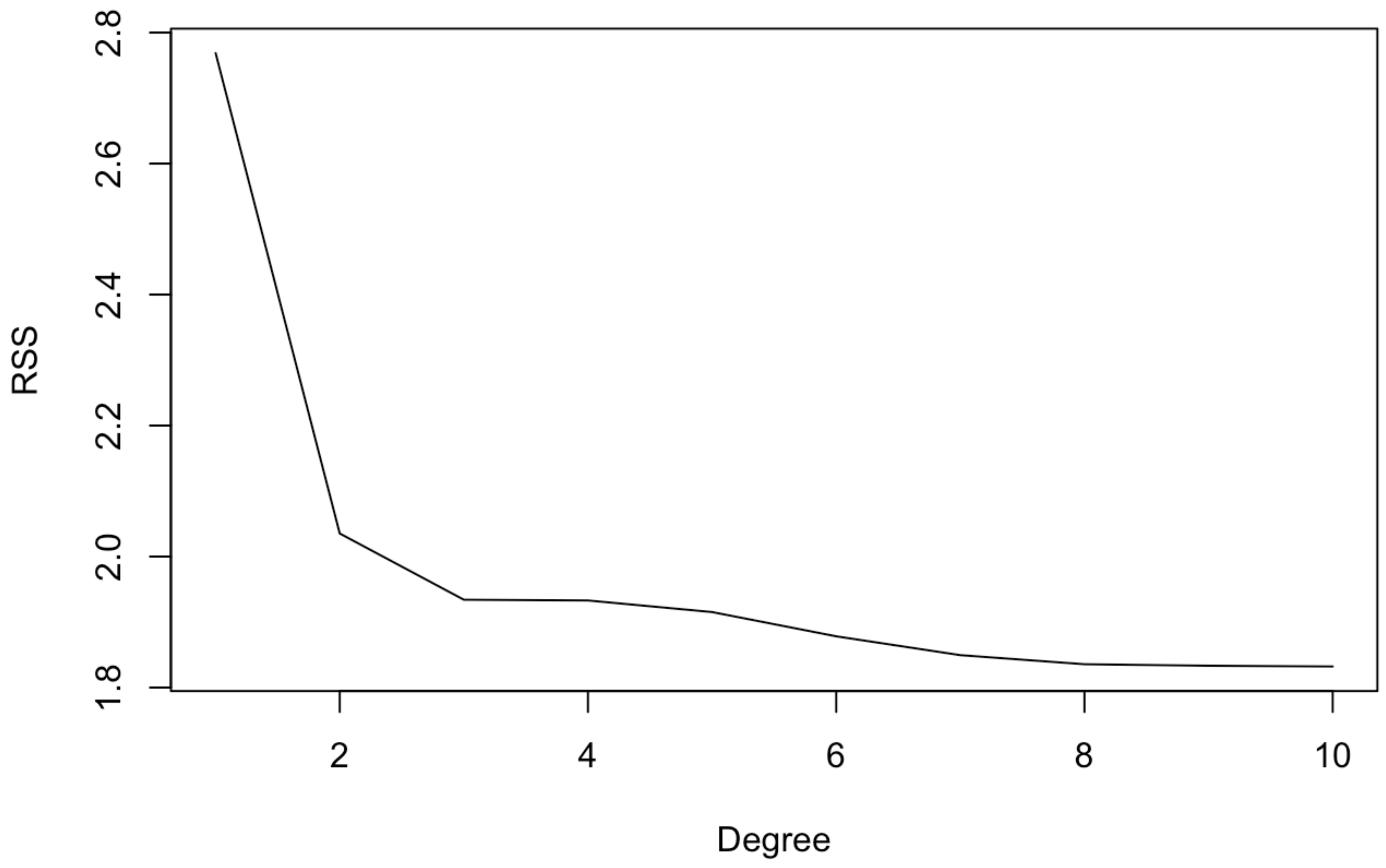
```
dislims=range(Boston$dis)
dis.grid=seq(from = dislims[1], to = dislims[2], by = 0.1)
preds=predict(fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, preds, col = "red", lwd = 2)
```



We could conclude that all polynomial terms are significant.

b

```
rss=rep(NA, 10)
for (i in 1:10) {
    fit=lm(nox ~ poly(dis, i), data = Boston)
    rss[i]=sum(fit$residuals^2)
}
plot(1:10, rss, xlab = "Degree", ylab = "RSS", type = "l")
```



It seems that the RSS decreases with the degree of the polynomial, and so is minimum for a polynomial of degree 10.

C

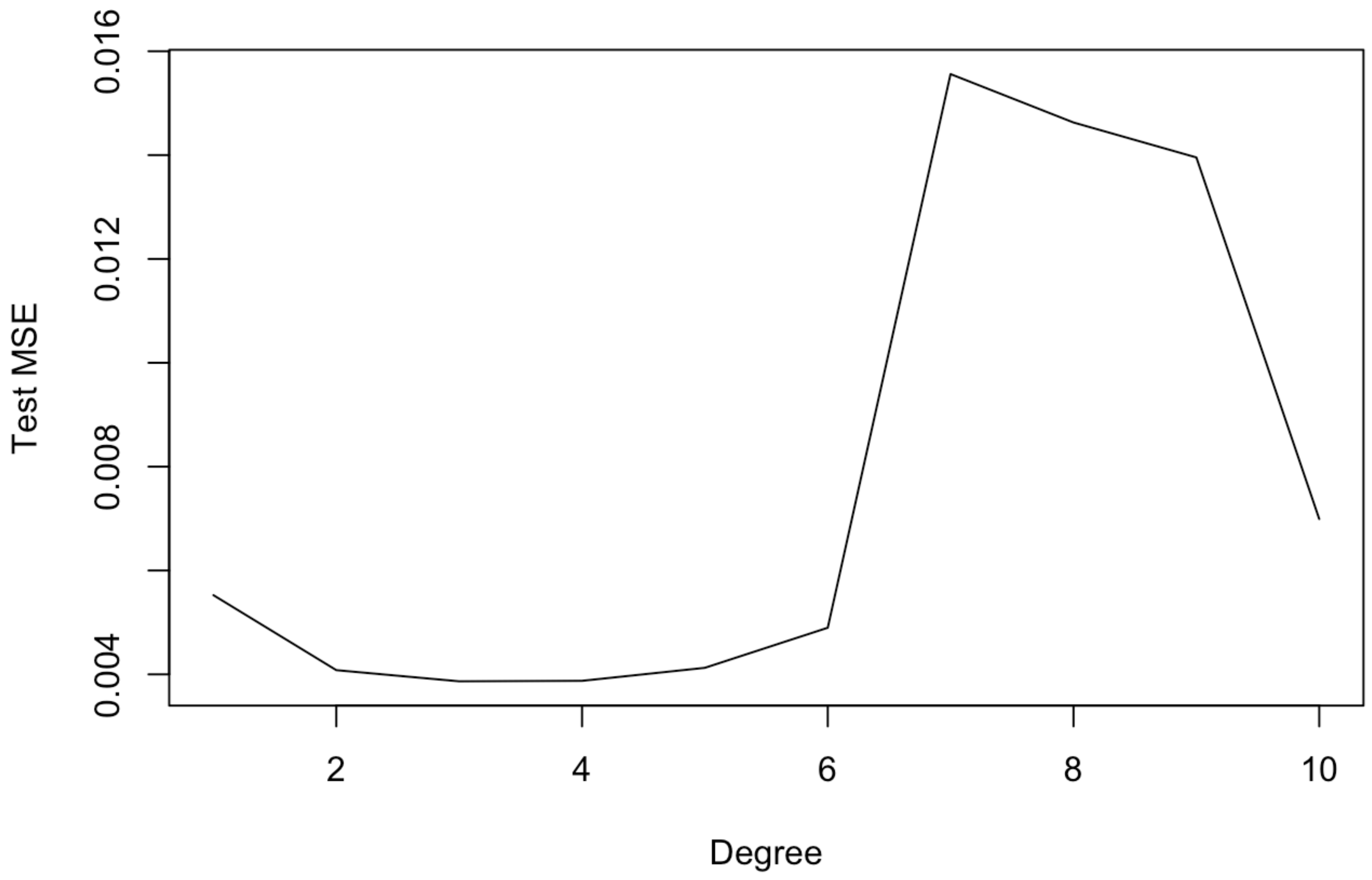
```
library(MASS)
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:mosaic':
##
##     logit
```

```
## The following object is masked from 'package:survival':
##
##     aml
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
deltas=rep(NA, 10)
for (i in 1:10) {
    fit=glm(nox ~ poly(dis, i), data = Boston)
    deltas[i]=cv.glm(Boston, fit, K = 10)$delta[1]
}
plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")
```



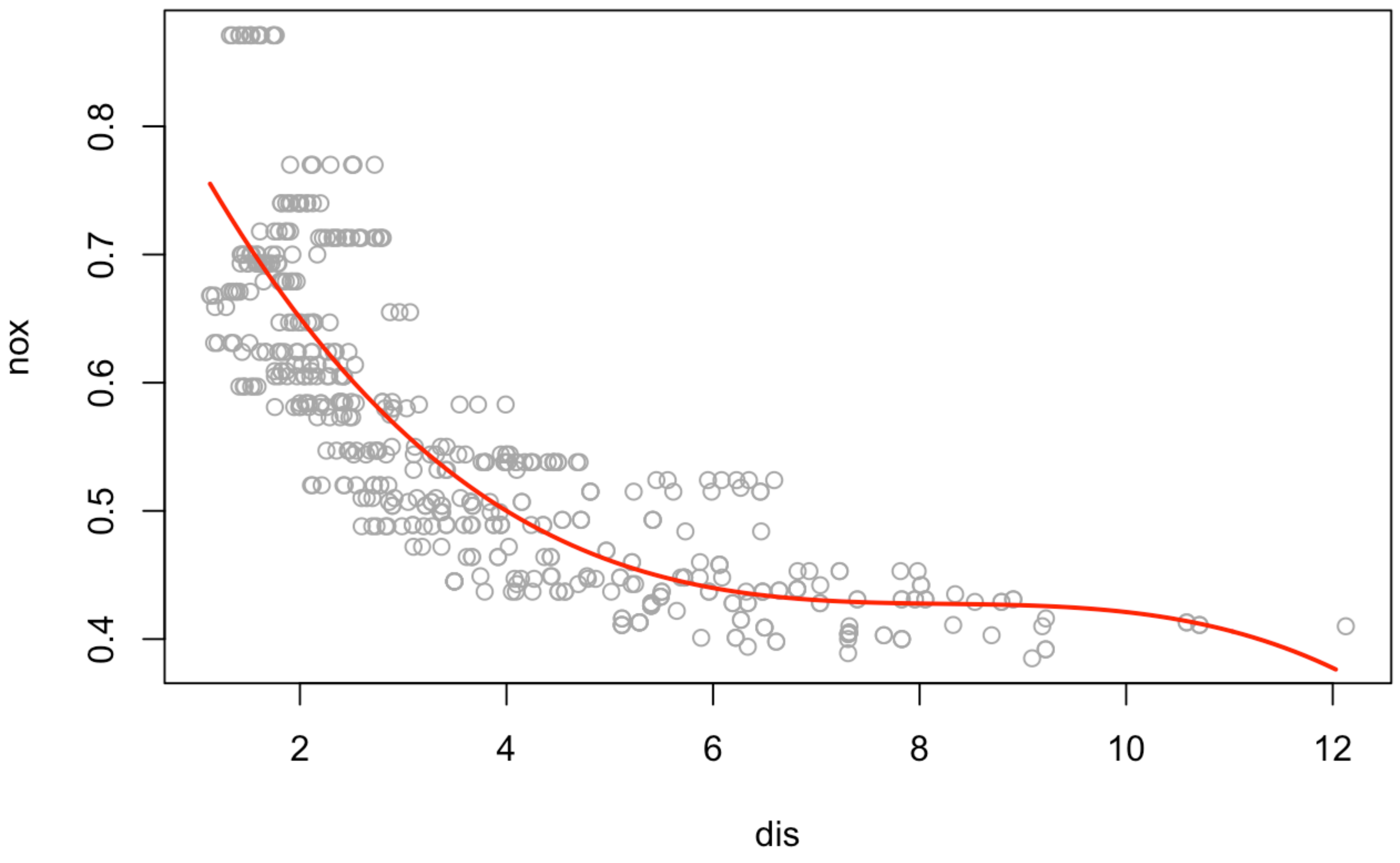
Above the picture, we could see that a polynomial of degree 4 minimizes the test MSE.

d

```
library(splines)
fit=lm(nox ~ bs(dis, knots = c(4, 7, 11)), data = Boston)
summary(fit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, knots = c(4, 7, 11)), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124567 -0.040355 -0.008702  0.024740  0.192920
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.73926    0.01331  55.537  < 2e-16 ***
## bs(dis, knots = c(4, 7, 11))1 -0.08861    0.02504  -3.539  0.00044 ***
## bs(dis, knots = c(4, 7, 11))2 -0.31341    0.01680 -18.658  < 2e-16 ***
## bs(dis, knots = c(4, 7, 11))3 -0.26618    0.03147  -8.459 3.00e-16 ***
## bs(dis, knots = c(4, 7, 11))4 -0.39802    0.04647  -8.565  < 2e-16 ***
## bs(dis, knots = c(4, 7, 11))5 -0.25681    0.09001  -2.853  0.00451 **
## bs(dis, knots = c(4, 7, 11))6 -0.32926    0.06327  -5.204 2.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06185 on 499 degrees of freedom
## Multiple R-squared:  0.7185, Adjusted R-squared:  0.7151
## F-statistic: 212.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

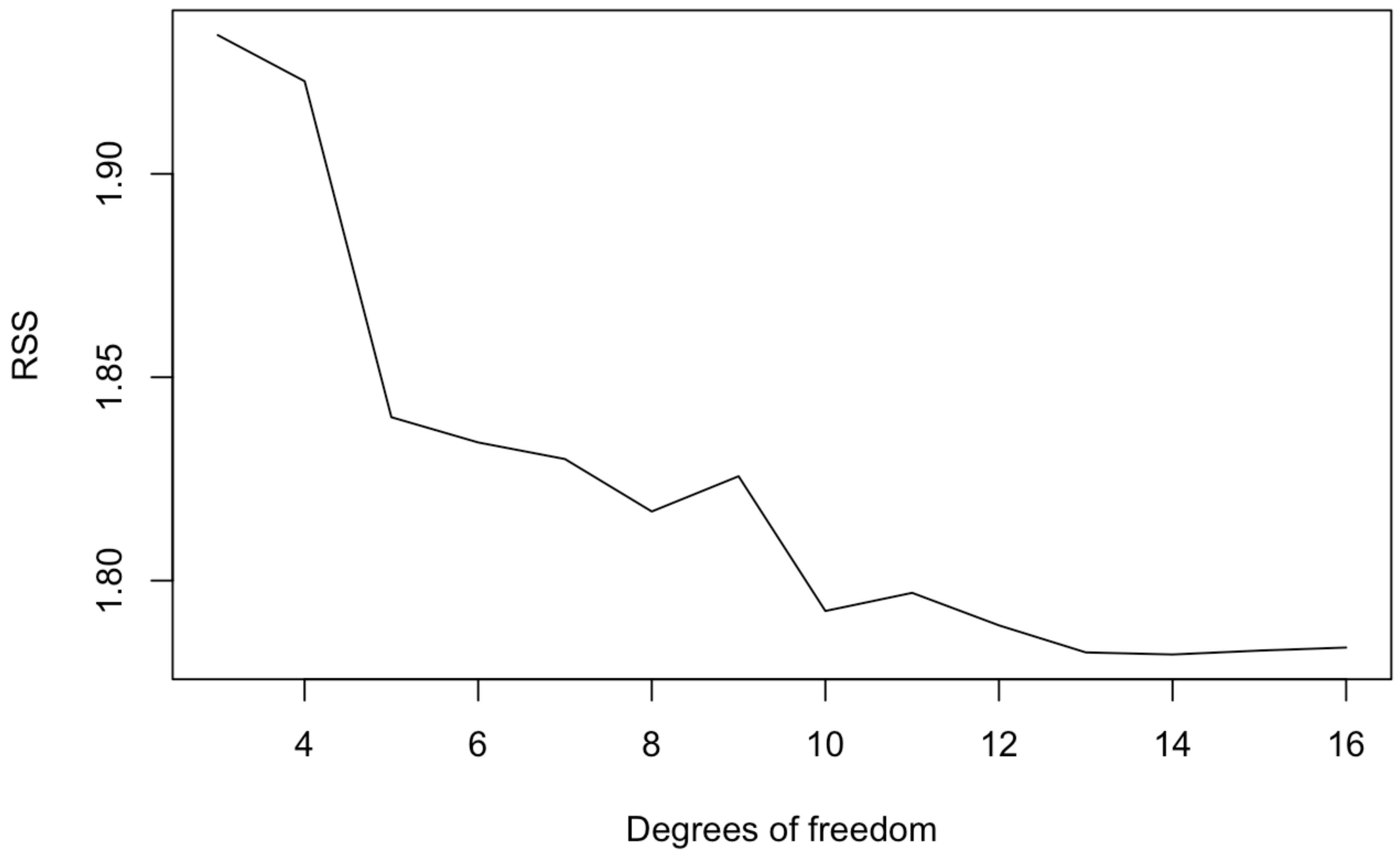
```
pred <- predict(fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, preds, col = "red", lwd = 2)
```



All the term are significant.

e

```r
rss=rep(NA, 16)
for (i in 3:16) {
    fit=lm(nox ~ bs(dis, df = i), data = Boston)
    rss[i]=sum(fit$residuals^2)
}
plot(3:16, rss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "RSS", type = "l")
```



We could see that RSS decreases until 14.

f

```
cv=rep(NA, 16)
for (i in 3:16) {
    fit=glm(nox ~ bs(dis, df = i), data = Boston)
    cv[i]=cv.glm(Boston, fit, K = 10)$delta[1]
}
```

```
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(3.2157, .Names =
## "50%"), : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(3.2157, .Names =
## "50%"), : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(2.40296666666667,
## 4.3549: some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(2.40296666666667,
## 4.3549: some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(2.4216, 4.4272), .Names
## = c("33.33333%", : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(2.4216, 4.4272), .Names
## = c("33.33333%", : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(2.087875, 3.19095,
## 4.97885: some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(2.087875, 3.19095,
## 4.97885: some 'x' values beyond boundary knots may cause ill-conditioned
## bases
## Warning in bs(dis, degree = 3L, knots = structure(c(2.10035, 3.1523,
## 5.2146: some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(2.10035, 3.1523,
## 5.2146: some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.94264, 2.59774,
## 3.9175, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.94264, 2.59774,
## 3.9175, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.9796, 2.70138,
## 3.93852, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.9796, 2.70138,
## 3.93852, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.82231666666667,
## 2.3772, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.82231666666667,
## 2.3772, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.8208, 2.3336,
## 3.0921, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.8208, 2.3336,
## 3.0921, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
## Warning in bs(dis, degree = 3L, knots = structure(c(1.78741428571429,
## 2.18085714285714, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.78741428571429,
## 2.18085714285714, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.78298571428571,
## 2.17271428571429, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.78298571428571,
## 2.17271428571429, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.733725, 2.0754,
## 2.4999, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.733725, 2.0754,
## 2.4999, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.751575, 2.10215,
## 2.4999, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.751575, 2.10215,
## 2.4999, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.66282222222222,
## 2.01021111111111, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.66282222222222,
## 2.01021111111111, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
## Warning in bs(dis, degree = 3L, knots = structure(c(1.66864444444444,
## 1.99707777777778, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.66864444444444,
## 1.99707777777778, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.62728, 1.92938,
## 2.25674, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.62728, 1.92938,
## 2.25674, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.61255, 1.9512,
## 2.26675, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.61255, 1.9512,
## 2.26675, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.64209090909091,
## 1.96261818181818, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.64209090909091,
## 1.96261818181818, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.61237272727273,
## 1.92314545454545, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.61237272727273,
## 1.92314545454545, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
## Warning in bs(dis, degree = 3L, knots = structure(c(1.55043333333333,
## 1.82023333333333, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.55043333333333,
## 1.82023333333333, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

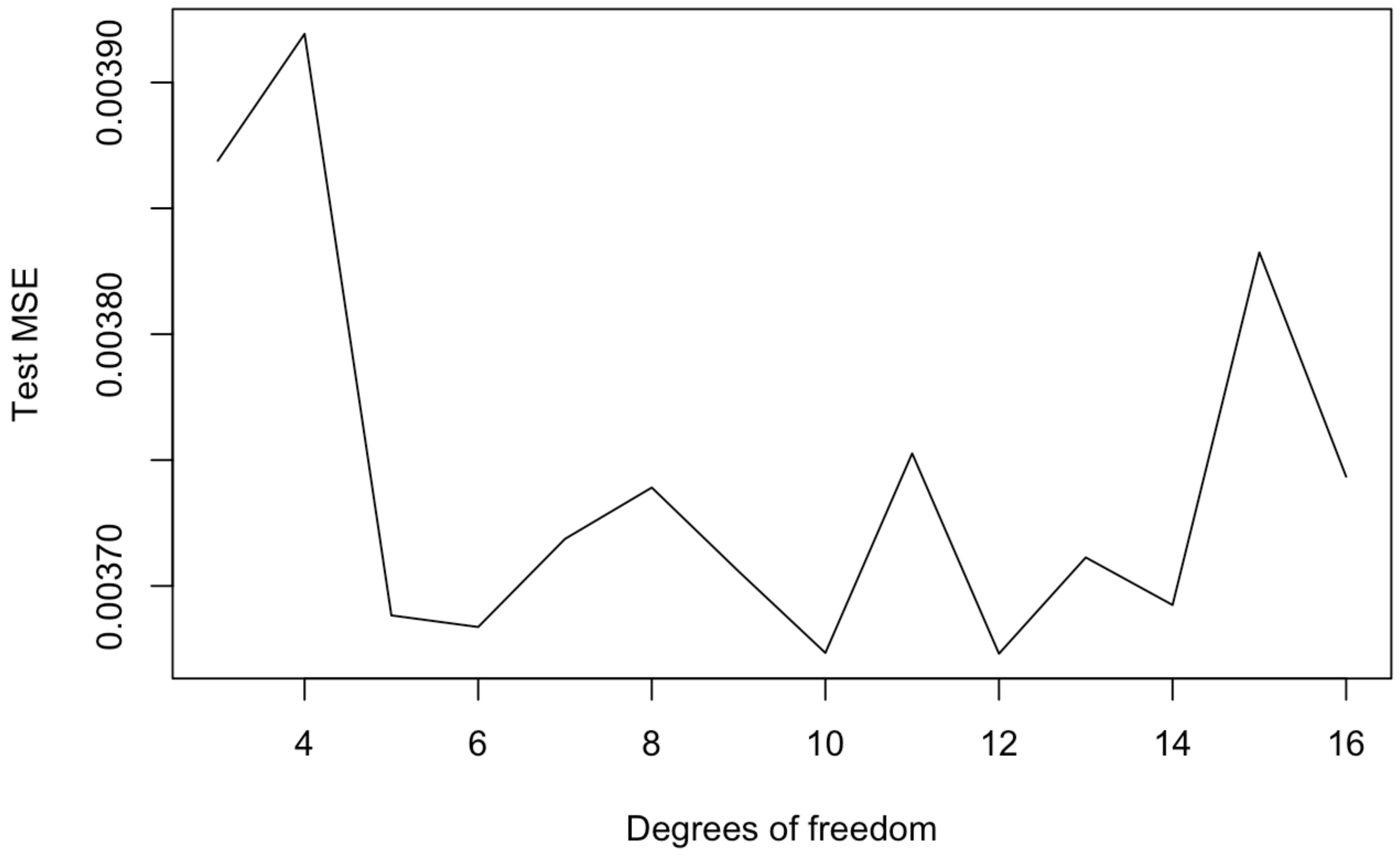
```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.58935,
## 1.86778333333333, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.58935,
## 1.86778333333333, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.52943846153846,
## 1.80235384615385, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.52943846153846,
## 1.80235384615385, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.58935384615385,
## 1.81647692307692, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.58935384615385,
## 1.81647692307692, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = structure(c(1.56255714285714,
## 1.80237142857143, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.56255714285714,
## 1.80237142857143, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
## Warning in bs(dis, degree = 3L, knots = structure(c(1.5846, 1.8172,
## 2.0001, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = structure(c(1.5846, 1.8172,
## 2.0001, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
plot(3:16, cv[-c(1, 2)], xlab = "Degrees of freedom", ylab = "Test MSE", type = "l")
```



Test MSE is minimum for 10 degrees of freedom.

HW6 Chp 8

Q9

a

```
set.seed(2022)
train=sample(1:nrow(OJ),800)

OJ.train=OJ[train,]
OJ.test=OJ[-train,]
```

b

```
library(tree)
OJ.tree=tree(Purchase~.,data=OJ.train)
summary(OJ.tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7608 = 602.6 / 792
## Misclassification error rate: 0.1725 = 138 / 800
```

The fitted tree has 8 terminal nodes and a training error rate of 0.173.

c

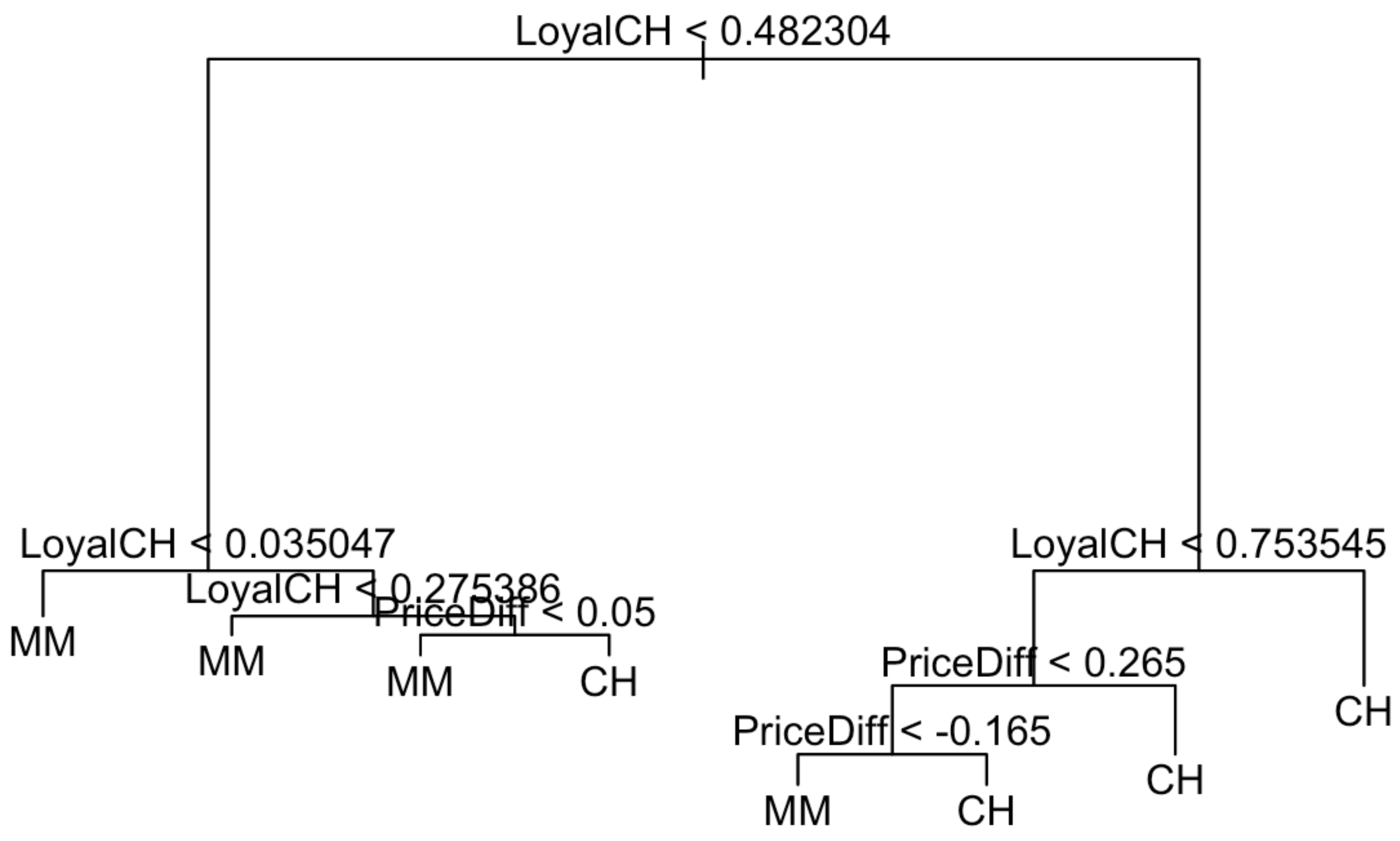
```
OJ.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1060.000 CH ( 0.62375 0.37625 )
##    2) LoyalCH < 0.482304 303  345.600 MM ( 0.25743 0.74257 )
##      4) LoyalCH < 0.035047 51    9.844 MM ( 0.01961 0.98039 ) *
##      5) LoyalCH > 0.035047 252  310.200 MM ( 0.30556 0.69444 )
##       10) LoyalCH < 0.275386 117  118.700 MM ( 0.20513 0.79487 ) *
##       11) LoyalCH > 0.275386 135  180.900 MM ( 0.39259 0.60741 )
##         22) PriceDiff < 0.05 54   57.210 MM ( 0.22222 0.77778 ) *
##         23) PriceDiff > 0.05 81  112.300 CH ( 0.50617 0.49383 ) *
##    3) LoyalCH > 0.482304 497  425.200 CH ( 0.84708 0.15292 )
##      6) LoyalCH < 0.753545 229  275.000 CH ( 0.71179 0.28821 )
##       12) PriceDiff < 0.265 146  198.400 CH ( 0.58219 0.41781 )
##         24) PriceDiff < -0.165 31   35.400 MM ( 0.25806 0.74194 ) *
##         25) PriceDiff > -0.165 115  145.900 CH ( 0.66957 0.33043 ) *
##       13) PriceDiff > 0.265 83   37.790 CH ( 0.93976 0.06024 ) *
##      7) LoyalCH > 0.753545 268   85.390 CH ( 0.96269 0.03731 ) *
```

We pick the node labelled 8, which is a terminal node because of the asterisk. The split criterion is LoyalCH < 0.035, the number of observations in that branch is 51 with a deviance of 9.84 and an overall prediction for the branch of MM.

d

```
plot(OJ.tree)
text(OJ.tree)
```



We may see that the most important indicator of `Purchase` appears to be `LoyalCH`, since the first branch differentiates the intensity of customer brand loyalty to CH.

e

```
library(knitr)
OJ.pred.train=predict(OJ.tree,OJ.train,type = 'class')
```

```
OJ.pred.test=predict(OJ.tree,OJ.test,type = 'class')
table(OJ.pred.test,OJ.test$Purchase)
```

```
##
## OJ.pred.test  CH  MM
##           CH 144  32
##           MM  10  84
```

```
table(OJ.pred.test,OJ.test$Purchase)/nrow(OJ.test)
```

```
##
## OJ.pred.test         CH         MM
##           CH 0.53333333 0.11851852
##           MM 0.03703704 0.31111111
```

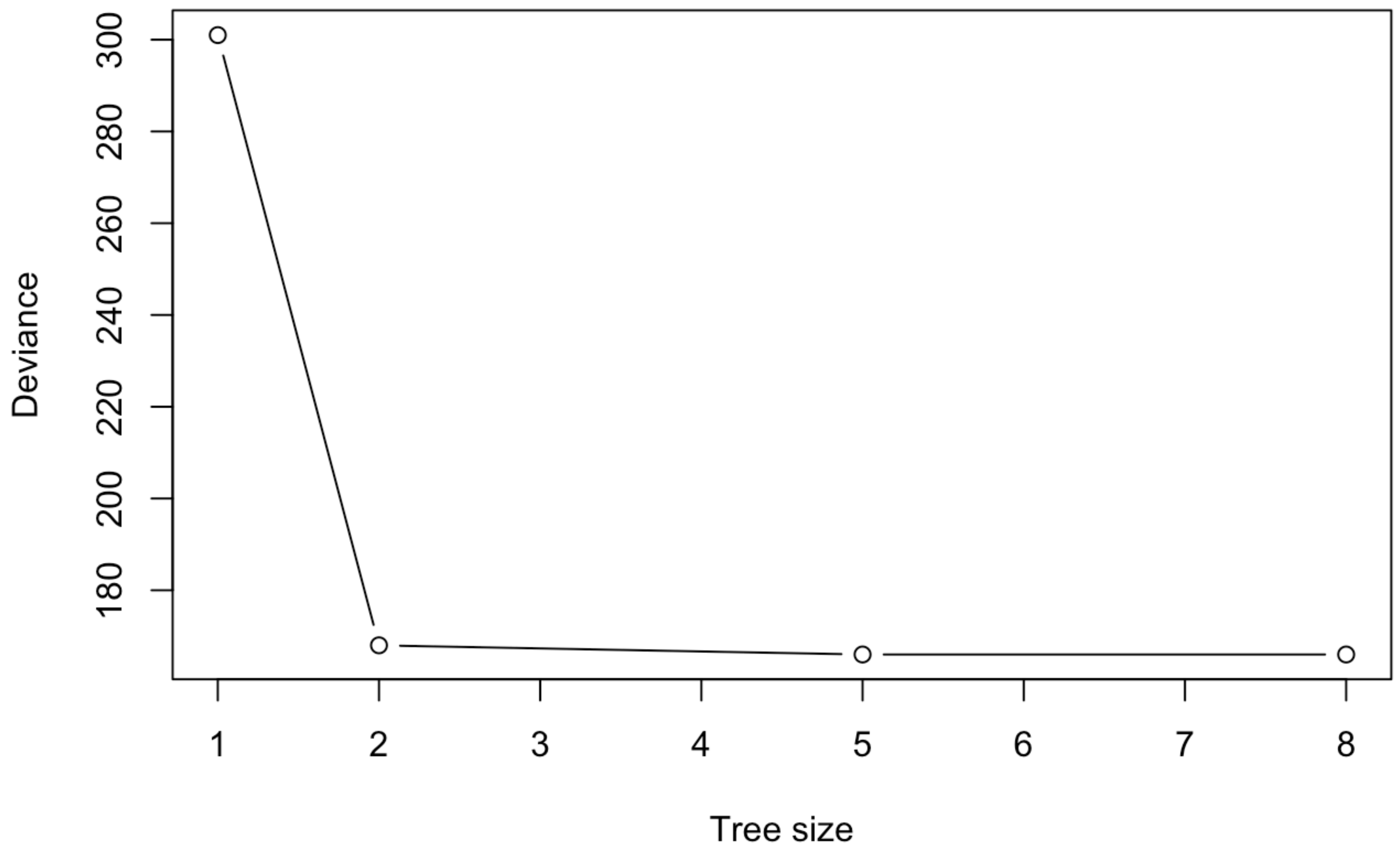
We may conclude that the test error rate is about 31%.

f

```
set.seed(2020)
OJ.tree.cv=cv.tree(OJ.tree,K = 10,FUN = prune.misclass)
```

g

```
plot(OJ.tree.cv$size, OJ.tree.cv$dev, type = "b", xlab = "Tree size", ylab = "Devianc
e")
```

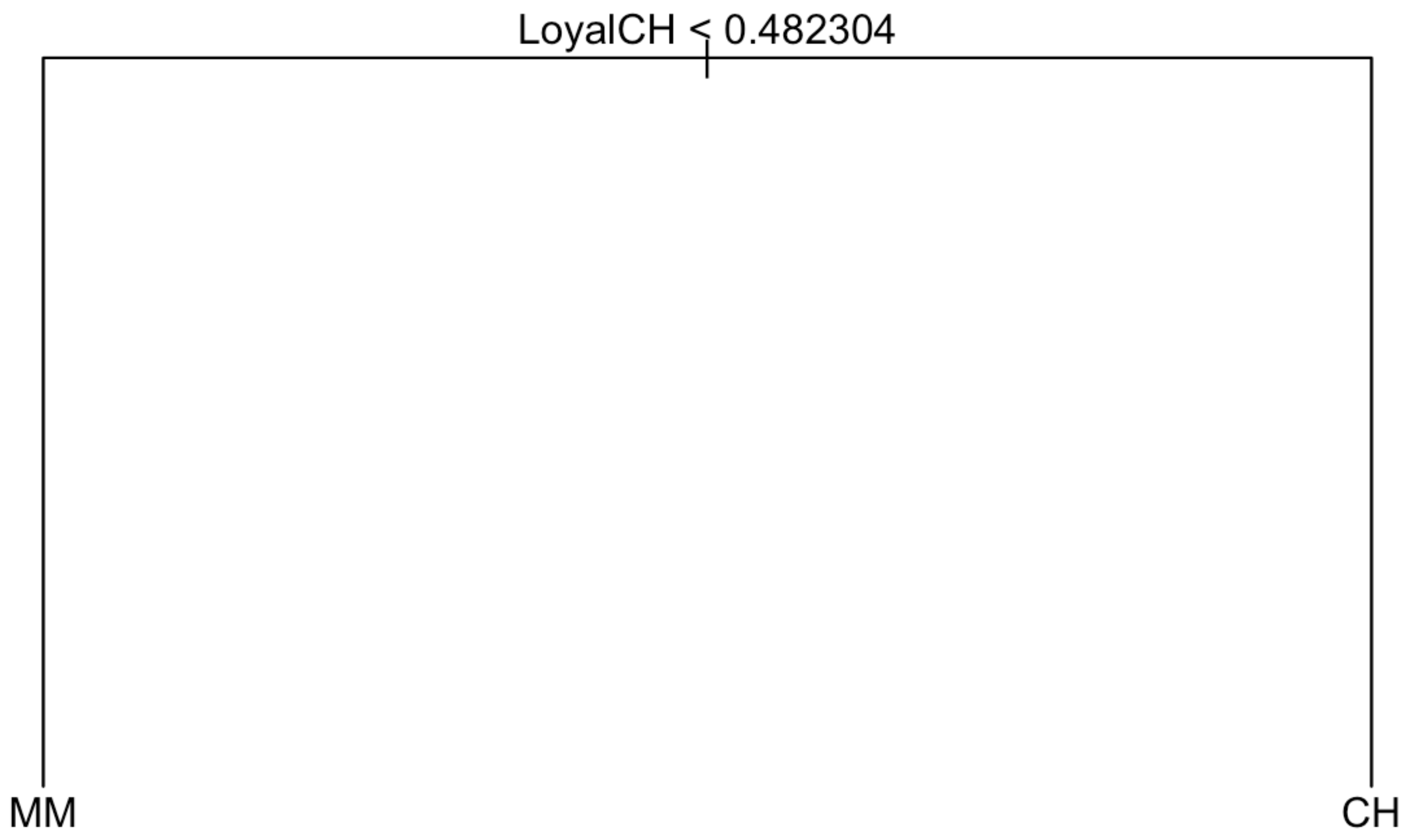


h

Above the picture, we could see that the optimal size is 2.

i

```
prune.oj=prune.misclass(OJ.tree,best = 2)

plot(prune.oj)
text(prune.oj, pretty = 0)
```



j

```
summary(prune.oj)
```

```
##
## Classification tree:
## snip.tree(tree = OJ.tree, nodes = 2:3)
## Variables actually used in tree construction:
## [1] "LoyalCH"
## Number of terminal nodes:  2
## Residual mean deviance:  0.9659 = 770.8 / 798
## Misclassification error rate: 0.1925 = 154 / 800
```

```
summary(OJ.tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7608 = 602.6 / 792
## Misclassification error rate: 0.1725 = 138 / 800
```

The misclassification error rate is slightly higher for the pruned tree (0.1825 vs 0.165).

k

```
prune.pred <- predict(prune.oj, OJ.test, type = "class")
table(prune.pred, OJ.test$Purchase)
```

```
##
## prune.pred  CH  MM
##         CH 138  34
##         MM  16  82
```

```
table(prune.pred,OJ.test$Purchase)/nrow(OJ.test)
```

```
##
## prune.pred         CH         MM
##         CH 0.51111111 0.12592593
##         MM 0.05925926 0.30370370
```

In this case, the pruning process increased the test error rate to about 30%, but it produced a way more interpretable tree.