HW6

Jon Griffith and Annabelle Cunningham

2025-04-21

```r
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.4.3
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.3
## Loading required package: Matrix
## Loaded glmnet 4.1-8
```

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.4.3
```

```r
library(pls)
```

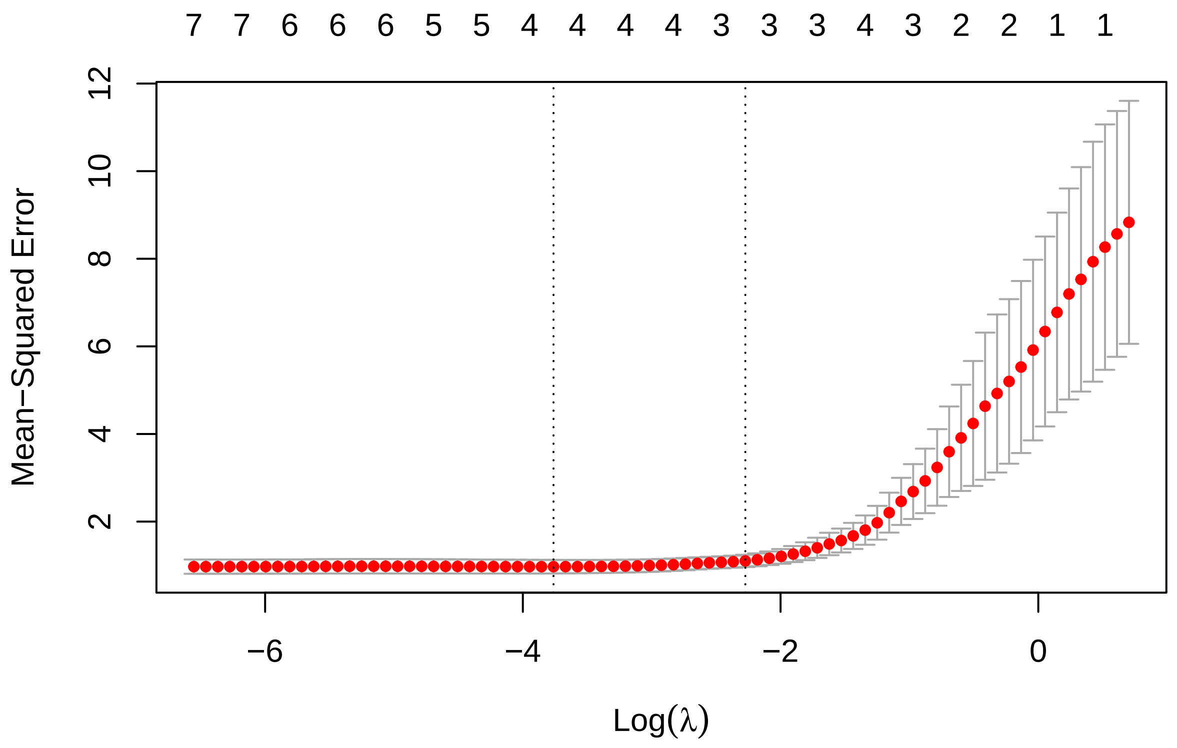
```
## Warning: package 'pls' was built under R version 4.4.3
##
## Attaching package: 'pls'
## The following object is masked from 'package:stats':
##
##     loadings
```

8

(e)

```r
set.seed(1)
# Generate data
x <- rnorm(100)
e <- rnorm(100)
Y <- 4 - 3*x - 2*x^2 + 1.25*x^3 + e

df <- data.frame(x,Y)
```

```r
set.seed(1)
cv.out <- cv.glmnet(poly(df$x,10,raw=TRUE), df$Y, alpha=1)

plot(cv.out)
```



```
lasso.mod <- glmnet(poly(df$x, 10, raw=TRUE), df$Y, alpha=1, lambda=cv.out$lambda.min)
predict(lasso.mod, type='coefficients', s=cv.out$lambda.min)
```

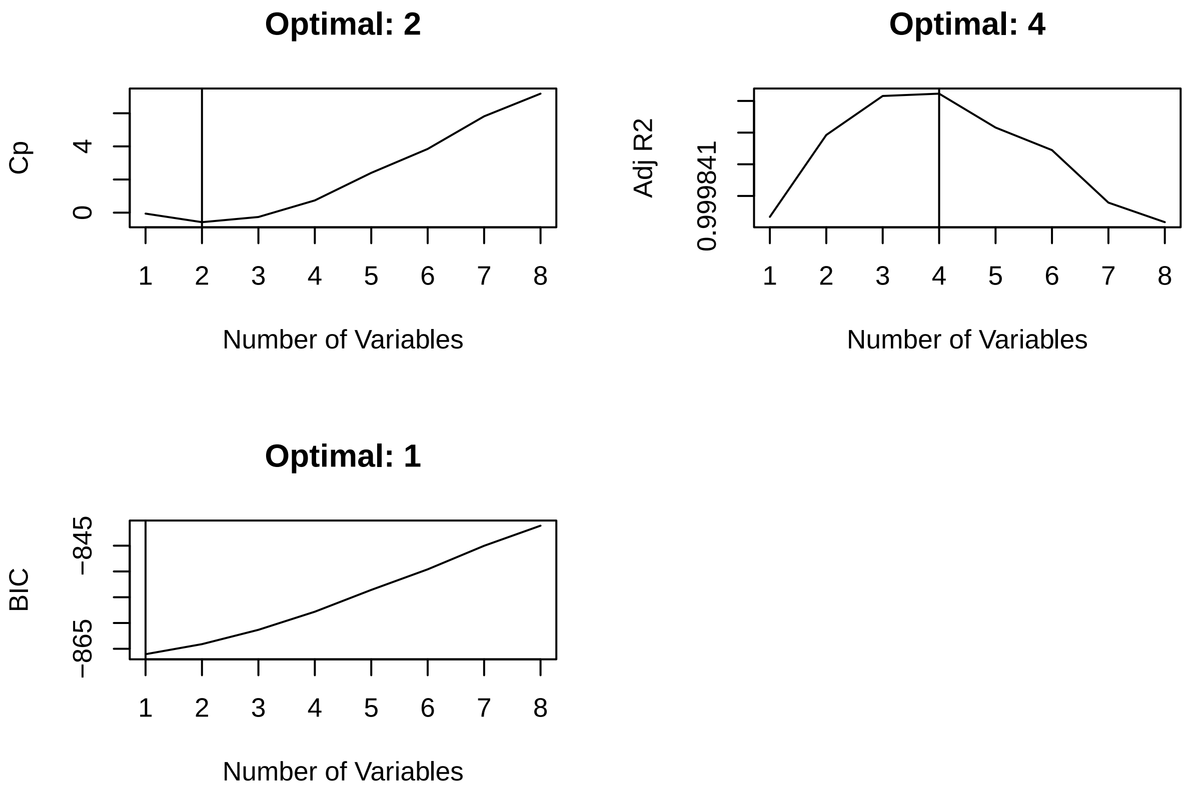
```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  4.0446480
## 1           -2.3121547
## 2           -2.1331883
## 3            0.5836176
## 4            .
## 5            0.1110500
## 6            .
## 7            .
## 8            .
## 9            .
## 10           .
```

Our resulting coefficient predictions based on the optimal lambda from CV come close to the true values for β_0, β_1, and β_2, while β_3 is about half of its actual value and β_5 is included even though there was no X^5 in the true equation. It still did a pretty good job of selecting close to the actual number of variables in the model and was only off by one.

(f)

```
set.seed(1)
df$Y <- 4 - 1.25*x^7 + e

bestsub.mod <- regsubsets(Y ~ poly(x, 10, raw=TRUE), data=df)

reg.summary <- summary(bestsub.mod)
```

```r
par(mfrow=c(2,2))
# PLot of cp vs num variables
plot(reg.summary$cp,
  xlab='Number of Variables',
  ylab='Cp',
  type='l',
  main=paste('Optimal:', which.min(reg.summary$cp)))
abline(v=which.min(reg.summary$cp))
# PLot of adjr2 vs num variables
plot(reg.summary$adjr2,
  xlab='Number of Variables',
  ylab='Adj R2',
  type='l',
  main=paste('Optimal:', which.max(reg.summary$adjr2)))
abline(v=which.max(reg.summary$adjr2))
# PLot of bic vs num variables
plot(reg.summary$bic,
  xlab='Number of Variables',
  ylab='BIC',
  type='l',
  main=paste('Optimal:', which.min(reg.summary$bic)))
abline(v=which.min(reg.summary$bic))
```



```r
coef(bestsub.mod, 1)
```

```
##             (Intercept) poly(x, 10, raw = TRUE)7
##                 3.95894                 -1.24923
```

```r
coef(bestsub.mod, 2)
```

```
##               (Intercept) poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)7
##                 4.0704904               -0.1417084               -1.2484448
```

```r
coef(bestsub.mod, 4)
```

```
##               (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2
##                 4.0762524                0.2914016               -0.1617671
## poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)7
##                -0.2526527               -1.2408662
```

```r
coef(bestsub.mod, 1)
```

```
##               (Intercept) poly(x, 10, raw = TRUE)7
##                   3.95894                 -1.24923
```

```r
coef(bestsub.mod, 2)
```

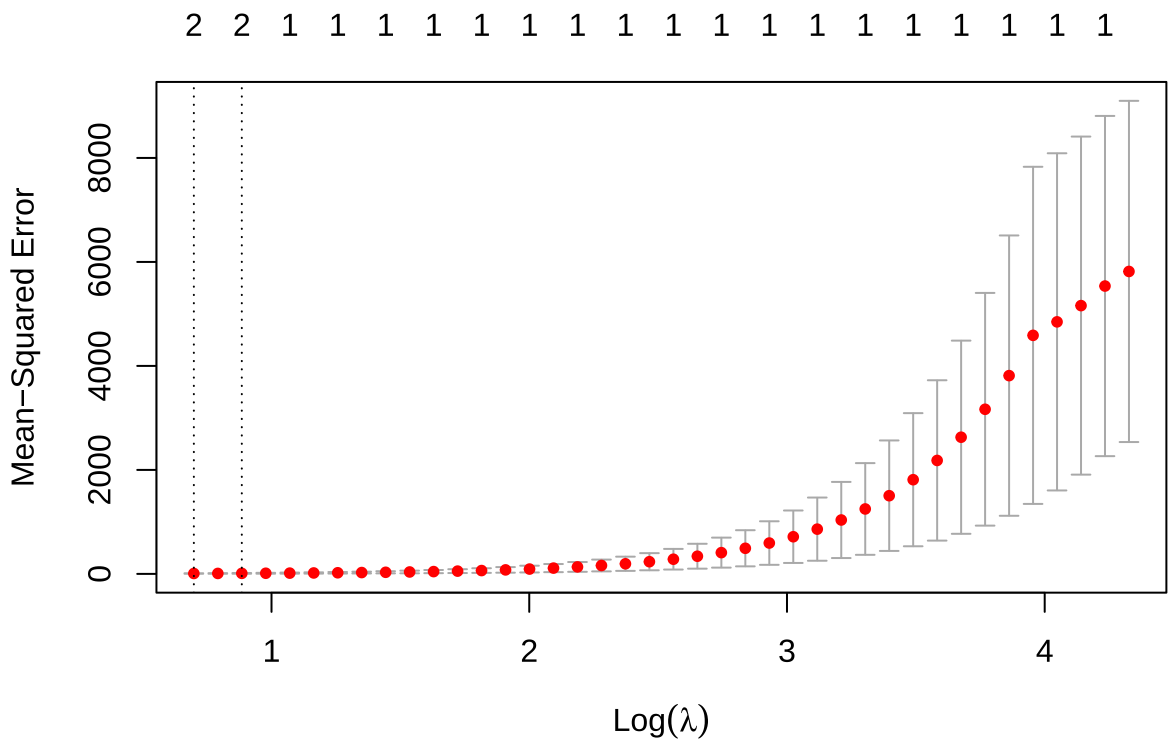
```
##               (Intercept) poly(x, 10, raw = TRUE)2 poly(x, 10, raw = TRUE)7
##                 4.0704904               -0.1417084               -1.2484448
```

```r
coef(bestsub.mod, 4)
```

```
##               (Intercept) poly(x, 10, raw = TRUE)1 poly(x, 10, raw = TRUE)2
##                 4.0762524                0.2914016               -0.1617671
## poly(x, 10, raw = TRUE)3 poly(x, 10, raw = TRUE)7
##                -0.2526527               -1.2408662
```

The best models as measured by BIC, CP, and Adj R^2 determined best subsets of (X^7), (X^2, X^7), and (X, X^2, X^3, and X^7), respectively. Each model included the predictor X^7 that corresponded to the true predictor, and the intercepts and coefficient for X^7 very closely approximated the true values for each. BIC did the best job but CP and Adj R^2 were nearly as good, adding only negligible effects of extra predictors.

```r
set.seed(1)
cv.out <- cv.glmnet(poly(df$x,10,raw=TRUE), df$Y, alpha=1)

plot(cv.out)
```



```
lasso.mod <- glmnet(poly(df$x, 10, raw=TRUE), df$Y, alpha=1, lambda=cv.out$lambda.min)
predict(lasso.mod, type='coefficients', s=cv.out$lambda.min)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  3.81888390
## 1             .
## 2             .
## 3             .
## 4             .
## 5            -0.01814104
## 6             .
## 7            -1.21252256
## 8             .
## 9             .
## 10            .
```

The optimal lambda and corresponding model for Lasso, per CV, includes an intercept value of 3.81 which very closely approximates the true intercept of 4, the X^5 predictor with a coefficient of -0.018, which is not in the true model but is negligible, and the X^7 predictor with a coefficient of -1.212, which very closely approximates the true coefficient of -1.25. The Lasso does a good job of replicating the true model and is about on par with the best subsets method.

9

(a)

```
df <- College
head(df)
```

```
##                               Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23         52
## Adelphi University               Yes 2186   1924    512        16         29
## Adrian College                   Yes 1428   1097    336        22         50
## Agnes Scott College              Yes  417    349    137        60         89
## Alaska Pacific University        Yes  193    146     55        16         44
## Albertson College                Yes  587    479    158        38         62
##                                F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University          2885         537     7440       3300   450
## Adelphi University                    2683        1227    12280       6450   750
## Adrian College                        1036          99    11250       3750   400
## Agnes Scott College                    510          63    12960       5450   450
## Alaska Pacific University              249         869     7560       4120   800
## Albertson College                      678          41    13500       3335   500
##                                Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University       2200  70       78      18.1          12   7041
## Adelphi University                 1500  29       30      12.2          16  10527
## Adrian College                     1165  53       66      12.9          30   8735
## Agnes Scott College                 875  92       97       7.7          37  19016
## Alaska Pacific University          1500  76       72      11.9           2  10922
## Albertson College                   675  67       73       9.4          11   9727
##                                Grad.Rate
## Abilene Christian University          60
## Adelphi University                    56
## Adrian College                        54
## Agnes Scott College                   59
## Alaska Pacific University             15
## Albertson College                     55
```

```
set.seed(1)
train <- sample(1:nrow(df), round(0.7*nrow(df)))
```

(b)

```
ols.mod <- lm(Apps ~ ., data=df[train,])
yhat <- predict(ols.mod, newdata = df[-train,])
MSE <- mean((yhat - df$Apps[-train])^2)
MSE
```

```
## [1] 1266407
```

(c)

```
set.seed(1)
train_split <- model.matrix(Apps ~ .-1, data=df[train,])
test_split <- model.matrix(Apps ~ .-1, data=df[-train,])
ridge.cv <- cv.glmnet(train_split, df$Apps[train], alpha=0)
yhat <- predict(ridge.cv, newx = test_split, s = ridge.cv$lambda.min)
MSE_ridge <- mean((yhat - df$Apps[-train])^2)
MSE_ridge
```

```
## [1] 1122733
```

```
ridge.coef <- predict(ridge.cv, type='coefficients', s=ridge.cv$lambda.min)[1:19,]
ridge.coef
```

```
##   (Intercept)     PrivateNo    PrivateYes        Accept        Enroll
## -1.835673e+03  2.319012e+02 -2.327310e+02  1.097370e+00  3.870257e-01
##     Top10perc     Top25perc   F.Undergrad   P.Undergrad      Outstate
##  2.656554e+01  2.809031e-02  5.802492e-02  3.525905e-02 -3.238312e-02
##    Room.Board         Books      Personal           PhD      Terminal
##  2.182992e-01  3.007454e-01 -3.084963e-02 -4.137527e+00 -4.295744e+00
##     S.F.Ratio   perc.alumni        Expend     Grad.Rate
##  1.473171e+01 -4.595790e+00  6.404545e-02  1.003295e+01
```

(d)

```
set.seed(1)
lasso.cv <- cv.glmnet(train_split, df$Apps[train], alpha=1)
yhat <- predict(lasso.cv, newx = test_split, s = lasso.cv$lambda.min)
MSE_lasso <- mean((yhat - df$Apps[-train])^2)
lasso.coef <- predict(lasso.cv, type='coefficients', s=lasso.cv$lambda.min)[1:19,]

MSE_lasso
```

```
## [1] 1259544
```

```
lasso.coef
```

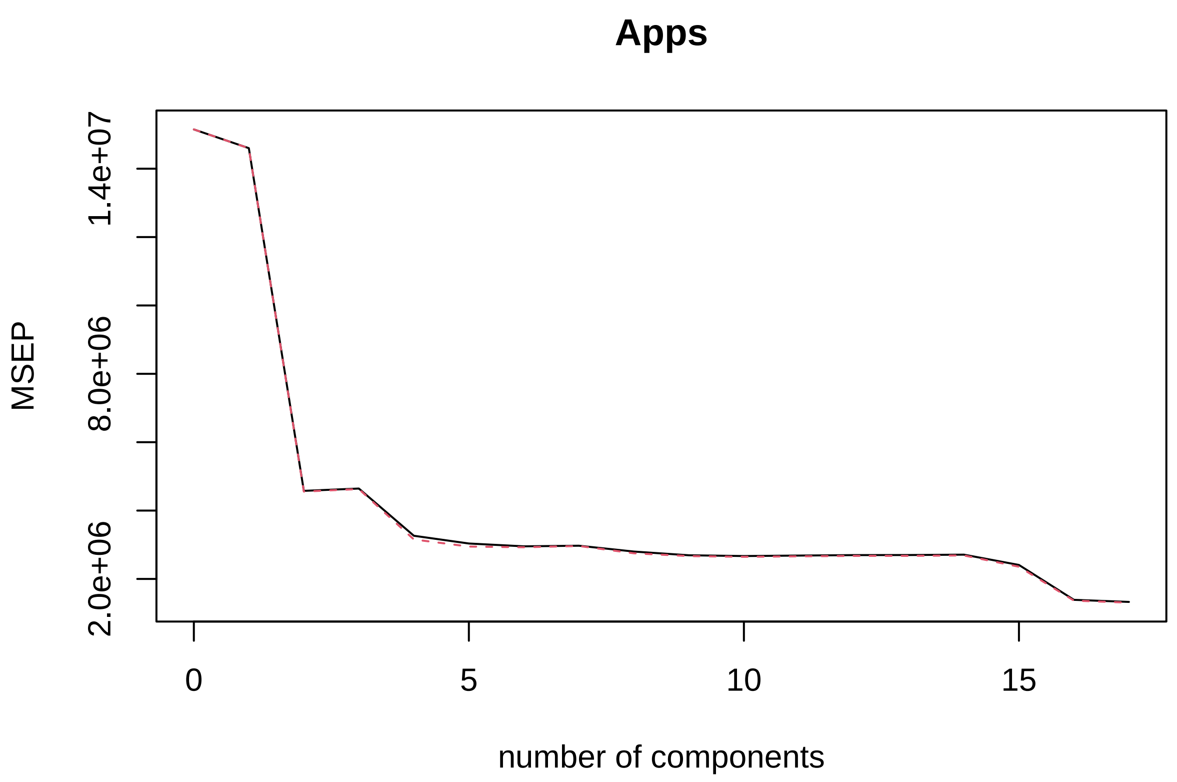
```
##   (Intercept)     PrivateNo    PrivateYes        Accept        Enroll
## -1.067871e+03  5.064555e+02 -1.271310e-09  1.706280e+00 -9.129561e-01
##     Top10perc     Top25perc   F.Undergrad   P.Undergrad      Outstate
##  5.220892e+01 -1.516972e+01  9.427128e-03  7.078118e-02 -8.550683e-02
##    Room.Board         Books      Personal           PhD      Terminal
##  1.603423e-01  2.622365e-01  0.000000e+00 -9.483740e+00 -8.895288e-02
##     S.F.Ratio   perc.alumni        Expend     Grad.Rate
##  1.530680e+01  1.523357e+00  5.921301e-02  6.948170e+00
```

We have only one predictor zeroed out and several others that are close to zero.

(e)

PCR

```
set.seed(1)
pcr.mod <- pcr(Apps ~ ., data=df[train,], scale=TRUE, validation='CV')
validationplot(pcr.mod, val.type='MSEP')
```



```
which.min(pcr.mod$validation$PRESS)
```

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

We have a minimum at the model with the same number of components as predictors, so we have $M = 17 = p$. This is equivalent to the OLS model, and so we will use this to find our test error equivalent to OLS.

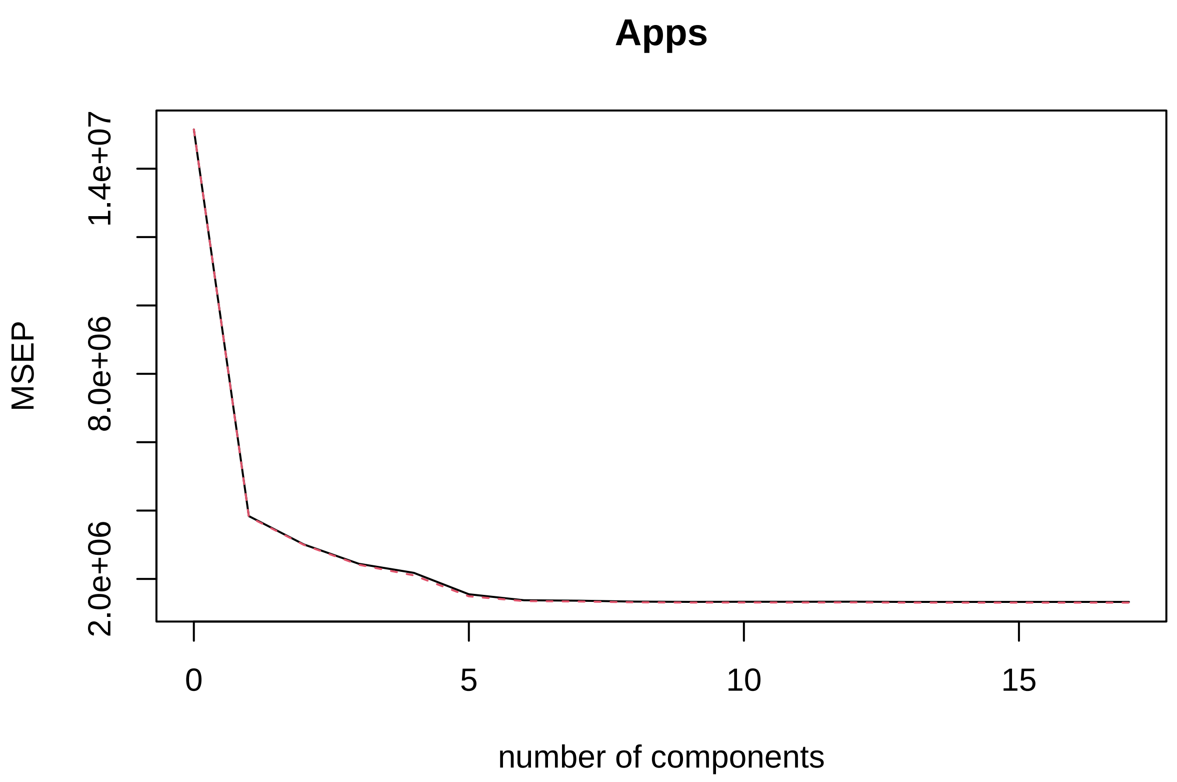
```
M <- 17
yhat <- predict(pcr.mod, df[-train,], ncomp = M)
MSE_pcr <- mean((yhat - df$Apps[-train])^2)

MSE_pcr
```

```
## [1] 1266407
```

(f)

PLS

```
set.seed(1)
pls.mod <- plsr(Apps ~ ., data=df[train,], scale=TRUE, validation='CV')
validationplot(pls.mod, val.type='MSEP')
```



```
which.min(pls.mod$validation$PRESS)
```

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

We again have a minimum at $M = 17$, which is equivalent to the OLS model, and so we will use this to find our test error equivalent to OLS.

```
M <- 17
yhat <- predict(pls.mod, df[-train,], ncomp = M)
MSE_pls <- mean((yhat - df$Apps[-train])^2)

MSE_pls
```

```
## [1] 1266407
```

(f)

```
cat('MSE OLS: ', MSE, '\n')
```

```
## MSE OLS:  1266407
```

```
cat('MSE Ridge: ', MSE_ridge, '\n')
```

```
## MSE Ridge:  1122733
```

```
cat('MSE Lasso: ', MSE_lasso, '\n')
```

```
## MSE Lasso:  1259544
```

```
cat('MSE PCR: ', MSE_pcr, '\n')
```

```
## MSE PCR:  1266407
```

```
cat('MSE PLS: ', MSE_pls, '\n')
```

```
## MSE PLS:  1266407
```

We achieved our best results from Ridge and Lasso while PCR and PLS just gave the OLS estimate. Out of Ridge and Lasso, Ridge did better.

11

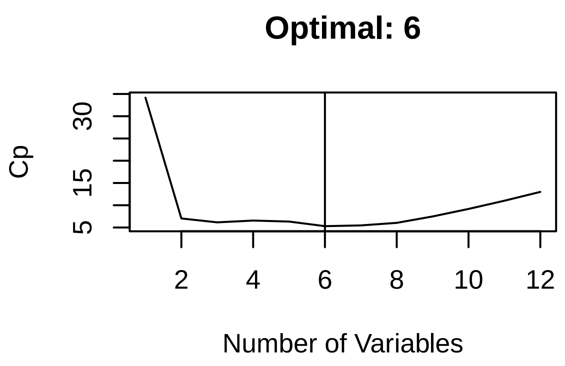
(a)

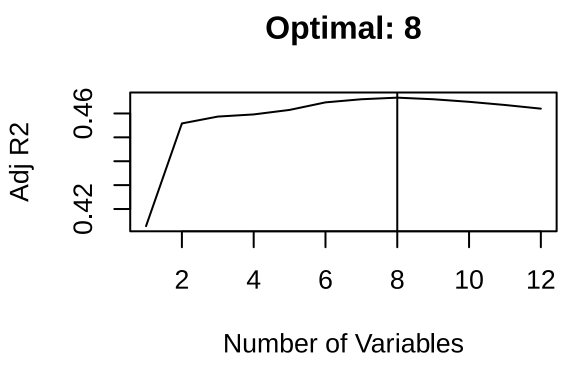
Create train/test splits

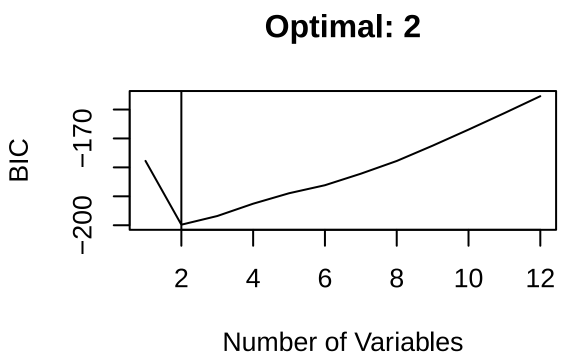
```
df <- Boston

set.seed(1)
train <- sample(1:nrow(df), round(0.7*nrow(df)))

train_split <- model.matrix(crim ~ .-1, data=df[train,])
test_split <- model.matrix(crim ~ .-1, data=df[-train,])
```

Best Subset Selection

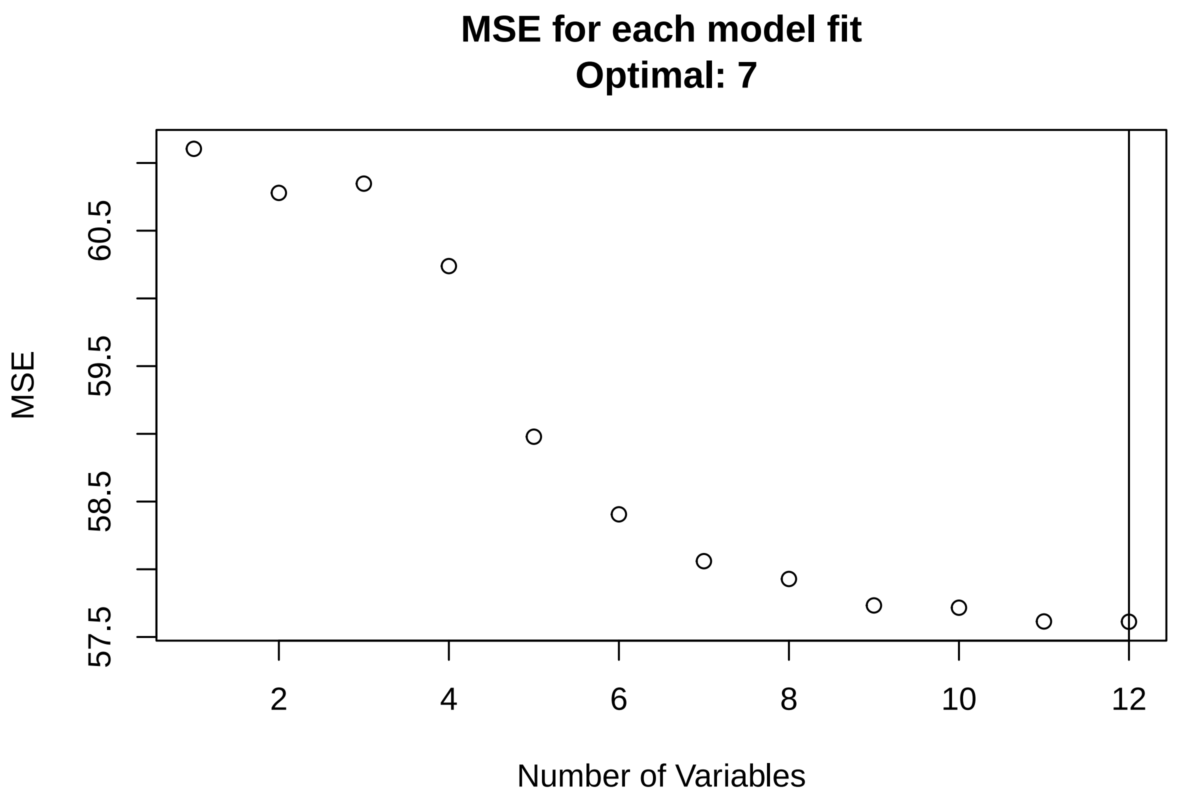
```
bestsub.mod <- regsubsets(crim ~ ., data=df[train,], nvmax=12)

reg.summary <- summary(bestsub.mod)

par(mfrow=c(2,2))
# PLot of cp vs num variables
plot(reg.summary$cp,
  xlab='Number of Variables',
  ylab='Cp',
  type='l',
  main=paste('Optimal:', which.min(reg.summary$cp)))
abline(v=which.min(reg.summary$cp))
# PLot of adjr2 vs num variables
plot(reg.summary$adjr2,
  xlab='Number of Variables',
  ylab='Adj R2',
  type='l',
  main=paste('Optimal:', which.max(reg.summary$adjr2)))
abline(v=which.max(reg.summary$adjr2))
# PLot of bic vs num variables
plot(reg.summary$bic,
  xlab='Number of Variables',
  ylab='BIC',
  type='l',
  main=paste('Optimal:', which.min(reg.summary$bic)))
abline(v=which.min(reg.summary$bic))
```







```r
test.mat <- model.matrix(crim ~ ., data=df[-train,])
val.errors <- rep(NA, 12)
for (i in 1:12){
  coefi <- coef(bestsub.mod, id=i)
  pred <- test.mat[,names(coefi)] %*% coefi
  val.errors[i] <- mean((df$crim[-train] - pred)^2)
}
plot(val.errors,
  main='MSE for each model fit\n Optimal: 7',
  xlab='Number of Variables',
  ylab='MSE')
abline(v = which.min(val.errors))
```



```
MSE_bestsub <- val.errors[2]
MSE_bestsub
```

```
## [1] 60.77902
```

Ridge Regression

```
ridge.cv <- cv.glmnet(train_split, df$crim[train], alpha=0)
yhat <- predict(ridge.cv, newx = test_split, s = ridge.cv$lambda.min)
MSE_ridge <- mean((yhat - df$crim[-train])^2)
MSE_ridge
```

```
## [1] 58.75168
```

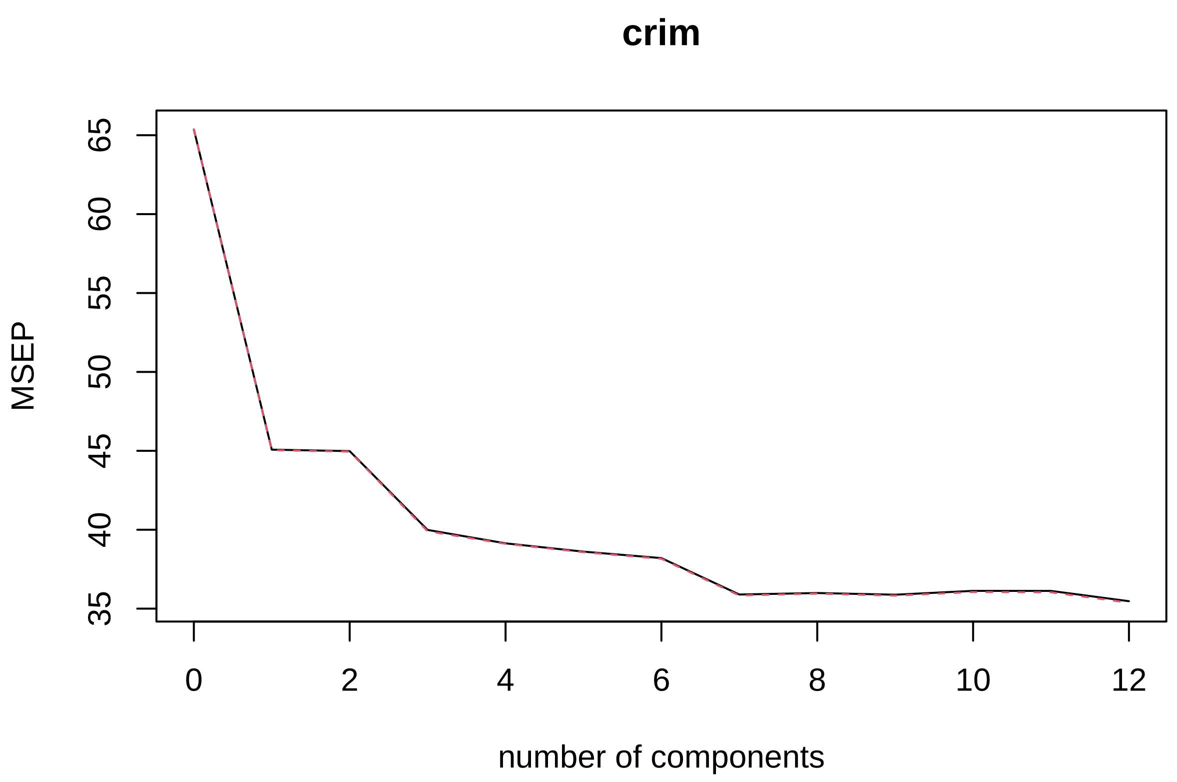
Lasso

```
lasso.cv <- cv.glmnet(train_split, df$crim[train], alpha=1)
yhat <- predict(lasso.cv, newx = test_split, s = lasso.cv$lambda.min)
MSE_lasso <- mean((yhat - df$crim[-train])^2)
MSE_lasso
```

```
## [1] 58.06509
```

PCR

```
set.seed(1)
pcr.mod <- pcr(crim ~ ., data=df[train,], scale=TRUE, validation='CV')
validationplot(pcr.mod, val.type='MSEP')
```



```r
which.min(pcr.mod$validation$PRESS)
```

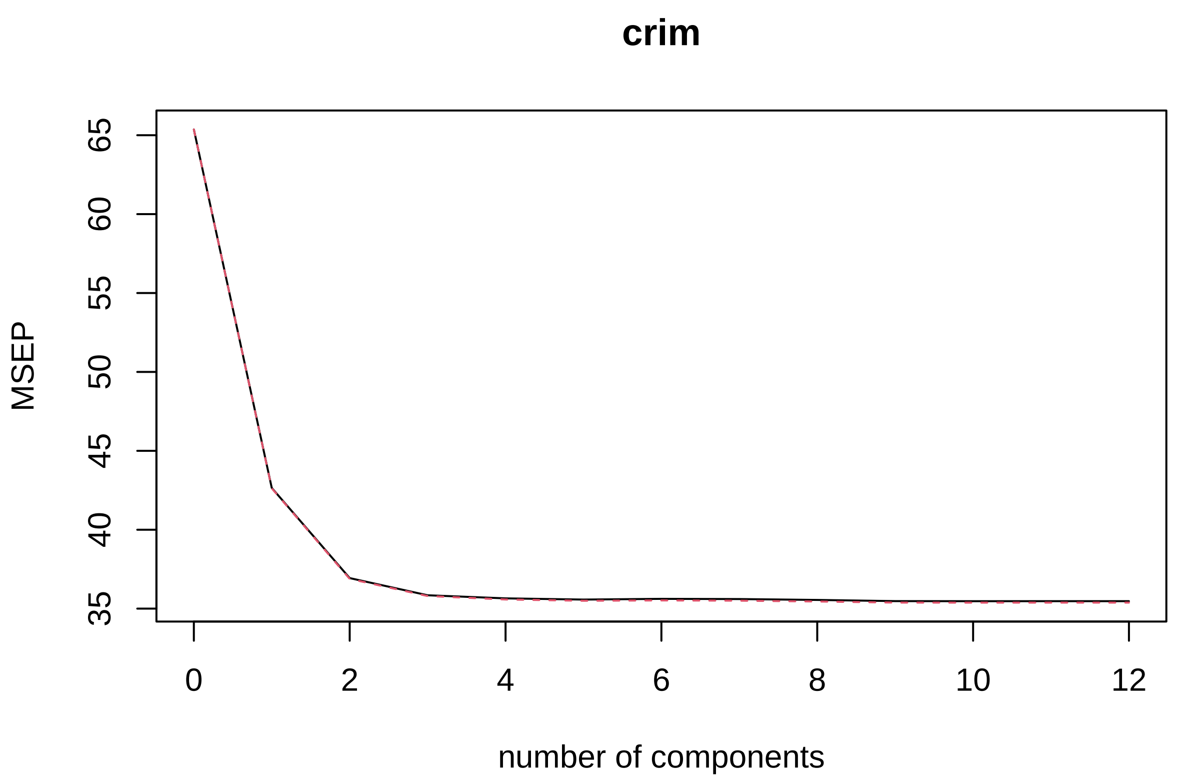
```
## [1] 12
```

```r
M <- 12
yhat <- predict(pcr.mod, df[-train,], ncomp = M)
MSE_pcr <- mean((yhat - df$crim[-train])^2)

MSE_pcr
```

```
## [1] 57.61252
```

PLS

```r
set.seed(1)
pls.mod <- plsr(crim ~ ., data=df[train,], scale=TRUE, validation='CV')
validationplot(pls.mod, val.type='MSEP')
```



```r
which.min(pls.mod$validation$PRESS)
```

```
## [1] 10
```

```r
M <- 10
yhat <- predict(pls.mod, df[-train,], ncomp = M)
MSE_pls <- mean((yhat - df$crim[-train])^2)

MSE_pls
```

```
## [1] 57.61217
```

(b)

```r
cat('MSE Best Subset: ', MSE_bestsub, '\n')
```

```
## MSE Best Subset:  60.77902
```

```r
cat('MSE Ridge: ', MSE_ridge, '\n')
```

```
## MSE Ridge:  58.75168
```

```r
cat('MSE Lasso: ', MSE_lasso, '\n')
```

```
## MSE Lasso:  58.06509
```

```r
cat('MSE PCR: ', MSE_pcr, '\n')
```

```
## MSE PCR:  57.61252
```

```r
cat('MSE PLS: ', MSE_pls, '\n')
```

```
## MSE PLS:  57.61217
```

We see that PCR and PLS perform the best out of best subset, ridge, lasso, PCR, and PLS, while PLS performs very slightly better than PCR. This is interesting because PCR is just OLS regression in this case since since the number of transformed predictors equals the number of original predictors ($M = p$). So we could just use the standard OLS model or use PLS with ten variables if we want a very slight improvement. Because PLS is less interpretable than OLS, we'd probably want to favor using OLS.

(c)

We chose PCR(OLS) as our final model. It includes all of the features in the dataset because that is how PCR works in general, it uses linear combinations of every feature for each transformed predictor. In this case, the PCR model is just the OLS model since $M = p$.