HW 6

Jared Andreatta

2025-04-21

Problem 2

a.

Option iii) is correct. By essentially constraining the estimate for $\lambda > 0$ by adding a penalty term on the ℓ_1 norm, this places further restriction on the OLS estimate. So this will reduce the variance of the estimate, but increase the bias of the estimate, so if the increase in bias is less than the decrease in variance, then it is effective in prediction accuracy improvement.

b.

Option iii) is also correct for this one. Similarly to the LASSO estimator, this estimator places a penalty term on the ℓ_2 norm for some $\lambda > 0$, placing further restrictions on the coefficient estimates.

c.

Option ii) is correct. Nonlinear models are less restrictive and may provide a better fit to data that exhibits nonlinear patterns, so it will reduce the bias of the estimate. So if the increase in variance is less than the decrease in bias, then it will be effective in prediction accuracy improvement.

Problem 3

a.

iv) is correct. As $s \to \infty$, then $\lambda \to 0$. As the Lagrangian multiplier becomes smaller, the estimate for β approaches the LS estimate, which will minimize the training RSS without being subjected to constraints.

b.

ii) is correct. The problem with LASSO regression is to minimize RSS by tuning the shrinkage parameter in order to minimize the RSS, which is a convex function with respect to the shrinkage parameter.

c.

iii) is correct. As we increase the shrinkage parameter s, the variance will will steadily increase, as the model becomes less restrictive and approaches the LS estimate.

d.

iv) is correct. This follows the same logic as part (c); as the variance of the estimate increases, then the bias of the estimate will decrease.

e.

v) is correct. The irreducible error is independent of any sort of measure of model flexibility, as it comes from exogenous factors, like measurement error.

Problem 4

a.

iii) is correct. As the Lagrangian multiplier becomes larger, the model becomes more restrictive than the LS estimate. Therefore, the training RSS will become larger as the model becomes more restrictive.

b.

ii) is correct. Similarly to problem 3 part (b), the optimization problem becomes finding the value of $\lambda > 0$ that minimizes the convex testing RSS function.

c.

iv) is correct. As λ increases, then the model becomes more restrictive and removes variability in the parameter estimates.

d.

iii) is correct. Conversely to variance, as the model becomes more restrictive, then the bias increases monotonically.

e.

v) is correct. This is for the same reason as problem 3 part (e).

Problem 8

e.

The coefficient estimates for the LASSO model are relatively close to their true values and does a relatively good job of variable selection. β^8 and β^{10} are nonzero, but small enough to be negligible in the context of the regression model.

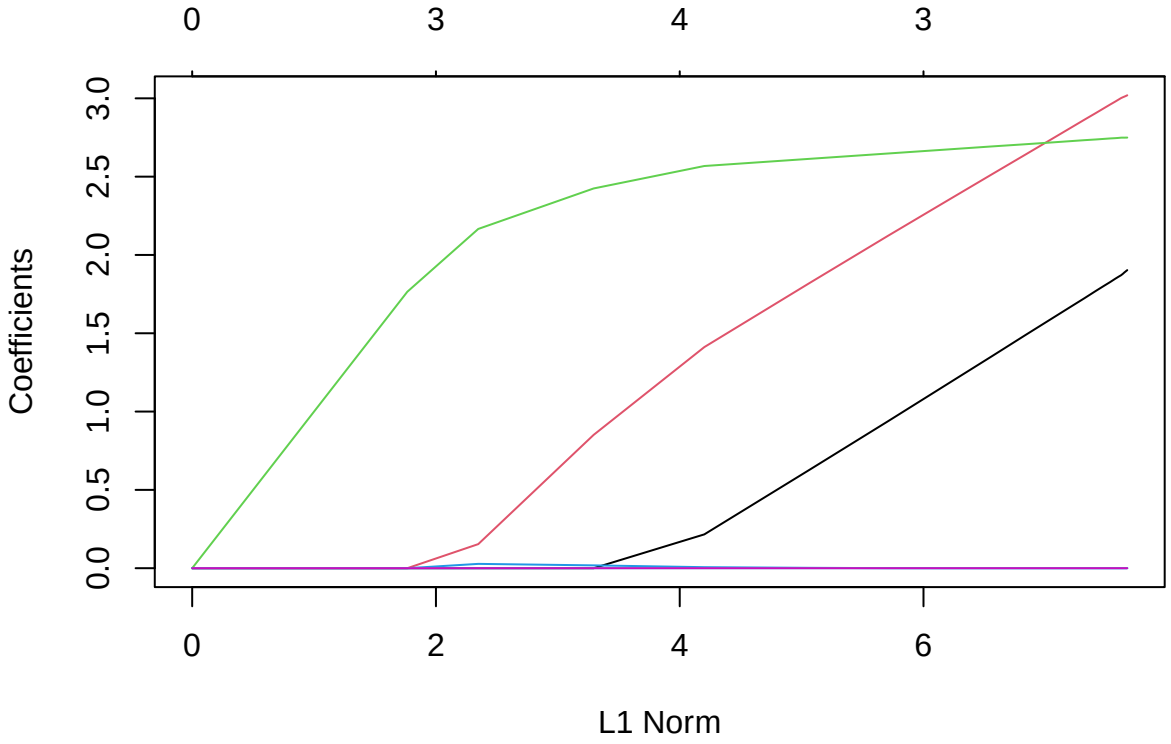
```
library(glmnet)
```

```
## Loading required package: Matrix
```

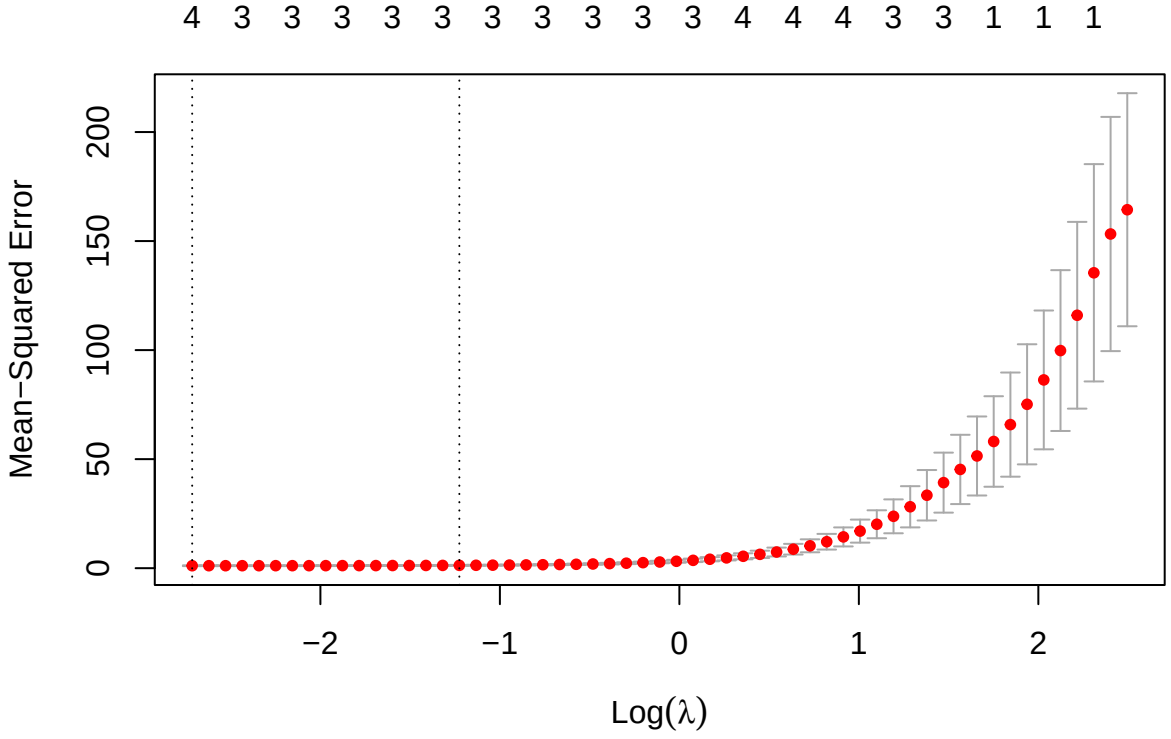
```
## Loaded glmnet 4.1-8
```

```
# DGP from Problem 8
set.seed(1234)
X <- rnorm(100)
eps <- rnorm(100)
Y <- 3 + 2*X + pi * X^2 + exp(1) * X^3 + eps

data <- data.frame(X,Y)

# LASSO model
grid <- 10^seq(10,-2, length = 100)
X.matrix <- model.matrix(Y~poly(X,10,raw=TRUE), data=data)
lasso_model <- glmnet(X.matrix, Y, lambda=grid, alpha=1)
plot(lasso_model)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
# CV
cv_lasso <- cv.glmnet(X.matrix, Y, alpha=1, standardize=TRUE, nfolds = 10)
plot(cv_lasso)
```



```r
opt_lambda <- cv_lasso$lambda.min
print(opt_lambda)
```

```
## [1] 0.06624601
```

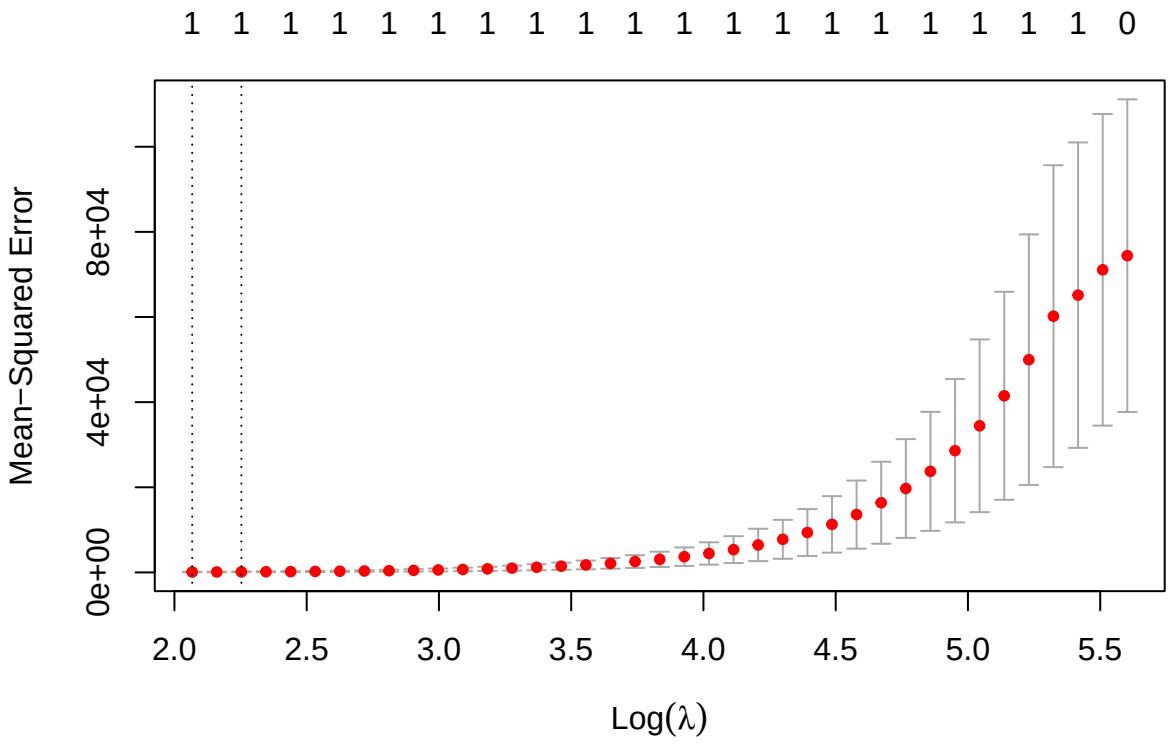
```r
# Predicting coefficients of full LASSO model with optimal lambda
predict(lasso_model, type = "coefficients", s = opt_lambda)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                3.166369e+00
## (Intercept)                .
## poly(X, 10, raw = TRUE)1   1.863658e+00
## poly(X, 10, raw = TRUE)2   2.994624e+00
## poly(X, 10, raw = TRUE)3   2.747757e+00
## poly(X, 10, raw = TRUE)4   .
## poly(X, 10, raw = TRUE)5   .
## poly(X, 10, raw = TRUE)6   .
## poly(X, 10, raw = TRUE)7   .
## poly(X, 10, raw = TRUE)8   5.505581e-06
## poly(X, 10, raw = TRUE)9   .
## poly(X, 10, raw = TRUE)10 7.523421e-09
```

f.

Both of the methods correctly "pick out" the correct coefficients, that is β_0 and β_7. The best subset selection method provides slightly more accurate estimates of the parameters.

```r
library(leaps)
# beta_7 model
set.seed(1234)
X2 <- rnorm(100)
eps <- rnorm(100)
Y2 <- pi + exp(1)*X^7 + eps

data <- data.frame(X2,Y2)

# Repeating LASSO procedure
grid <- 10^seq(10,-2, length = 100)
X2.matrix <- model.matrix(Y2~poly(X2,10,raw=TRUE), data=data)
lasso_model2 <- glmnet(X2.matrix, Y2, lambda=grid, alpha=1)

cv_lasso2 <- cv.glmnet(X2.matrix, Y2, alpha=1, standardize=TRUE, nfolds = 10)
plot(cv_lasso2)
```




```
opt_lambda2 <- cv_lasso2$lambda.min
print(opt_lambda2)
```

```
## [1] 7.901808
```

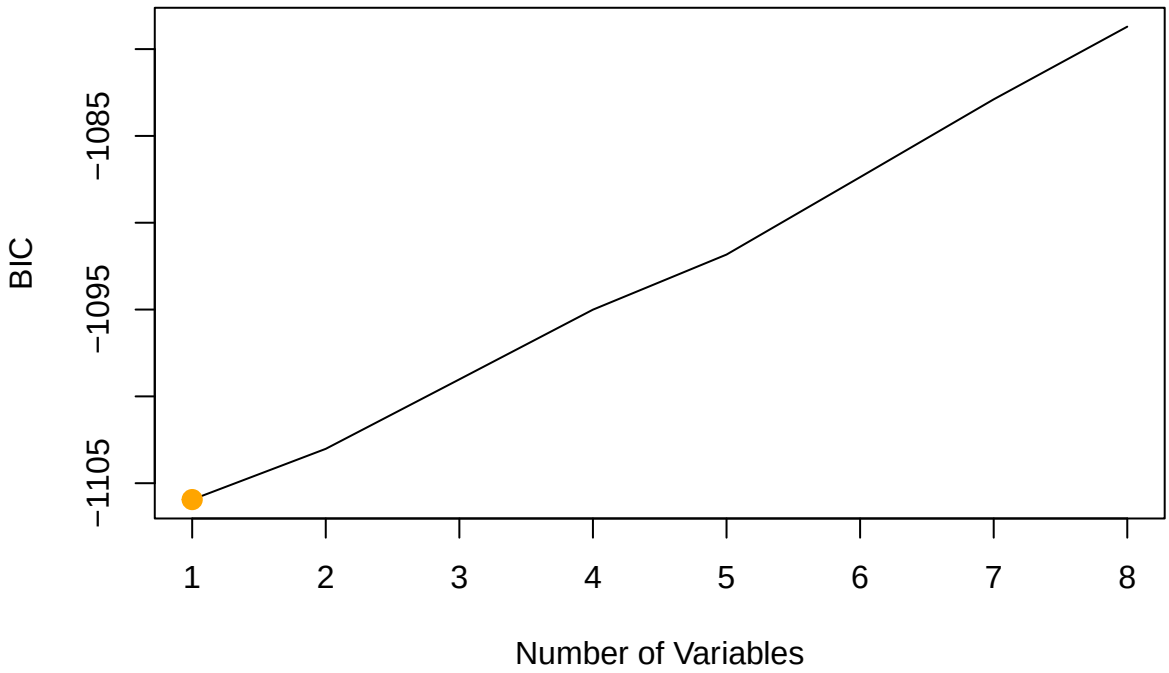
```
predict(lasso_model2, type = "coefficients", s = opt_lambda2)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                3.925728
## (Intercept)                .
## poly(X2, 10, raw = TRUE)1  .
## poly(X2, 10, raw = TRUE)2  .
## poly(X2, 10, raw = TRUE)3  .
## poly(X2, 10, raw = TRUE)4  .
## poly(X2, 10, raw = TRUE)5  .
## poly(X2, 10, raw = TRUE)6  .
## poly(X2, 10, raw = TRUE)7  2.638953
## poly(X2, 10, raw = TRUE)8  .
## poly(X2, 10, raw = TRUE)9  .
## poly(X2, 10, raw = TRUE)10 .
```

```
# Repeating best subset selection procedure
regfit <- regsubsets(Y2~poly(X2,10,raw=TRUE), data=data)

regfit.summary <- summary(regfit)
which.min(regfit.summary$bic) # BIC
```

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

```
plot(regfit.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
points(1, regfit.summary$bic[1], col = "orange", cex = 2, pch = 20)
```




```
coef(regfit, 1)
```

```
##               (Intercept) poly(X2, 10, raw = TRUE)7
##                  3.183698                  2.718190
```

Problem 9

a and b.

```
library(ISLR2)
# a.
set.seed(13123)
train_idx <- sample(c(TRUE, FALSE), nrow(College), replace=TRUE, prob=c(0.8,0.2))
train <- College[train_idx, ]
test  <- College[!train_idx, ]

# b. Test MSE = 1739432
ls.fit <- lm(Apps ~ ., data=train)
pred <- predict(ls.fit, newdata=test)
mean((test$Apps - pred)^2)
```

```
## [1] 1739432
```

c.

```
# c. Optimal lambda = 381.4014. Test MSE = 1624530. Test MSE for Ridge is
# 1624530, which is better than the OLS fit
set.seed(1212)
X.train.matrix <- model.matrix(Apps~.,data=train)[,-1]
Y.train <- train$Apps

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

```
cv.ridge.fit <- cv.glmnet(X.train.matrix, Y.train, standardize=TRUE,
                          nfolds=10, alpha=0)
best_lambda <- cv.ridge.fit$lambda.min
best_lambda
```

```
## [1] 381.4014
```

```
X.test.matrix <- model.matrix(Apps ~ ., data = test)[,-1]
preds <- predict(cv.ridge.fit, s = best_lambda, newx = X.test.matrix)
test_mse <- mean((test$Apps - preds)^2)
print(test_mse)
```

```
## [1] 1624530
```

d.

```
# d. The LASSO fit. Best lambda = 2.035424 and Test MSE is 1728065 All but 2
# coefficients are nonzero (Books, Terminal).
set.seed(1212)
X.train.matrix <- model.matrix(Apps~.,data=train)[,-1]
Y.train <- train$Apps
grid <- 10^seq(10,-2, length = 100)
cv.lasso.fit <- cv.glmnet(X.train.matrix, Y.train, standardize=TRUE,
                          nfolds=10, alpha=1) #alpha=1 for LASSO
best_lambda <- cv.lasso.fit$lambda.min
best_lambda
```

```
## [1] 2.035424
```

```
X.test.matrix <- model.matrix(Apps ~ ., data = test)[,-1]
preds <- predict(cv.lasso.fit, s = best_lambda, newx = X.test.matrix)
test_mse <- mean((test$Apps - preds)^2)
print(test_mse)
```

```
## [1] 1728065
```

```
predict(cv.lasso.fit, type = "coefficients", s = best_lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) -296.44846463
## PrivateYes  -441.22866637
## Accept         1.63449220
## Enroll        -1.07167258
## Top10perc     37.59316725
## Top25perc     -5.31345278
## F.Undergrad    0.07665350
## P.Undergrad   -0.01806830
## Outstate      -0.09597427
## Room.Board     0.17045271
## Books          .
## Personal      -0.01542962
## PhD          -10.04219748
## Terminal       .
## S.F.Ratio      6.68310148
## perc.alumni   -2.43386729
```

```
## Expend          0.06684752
## Grad.Rate       6.77693756
```

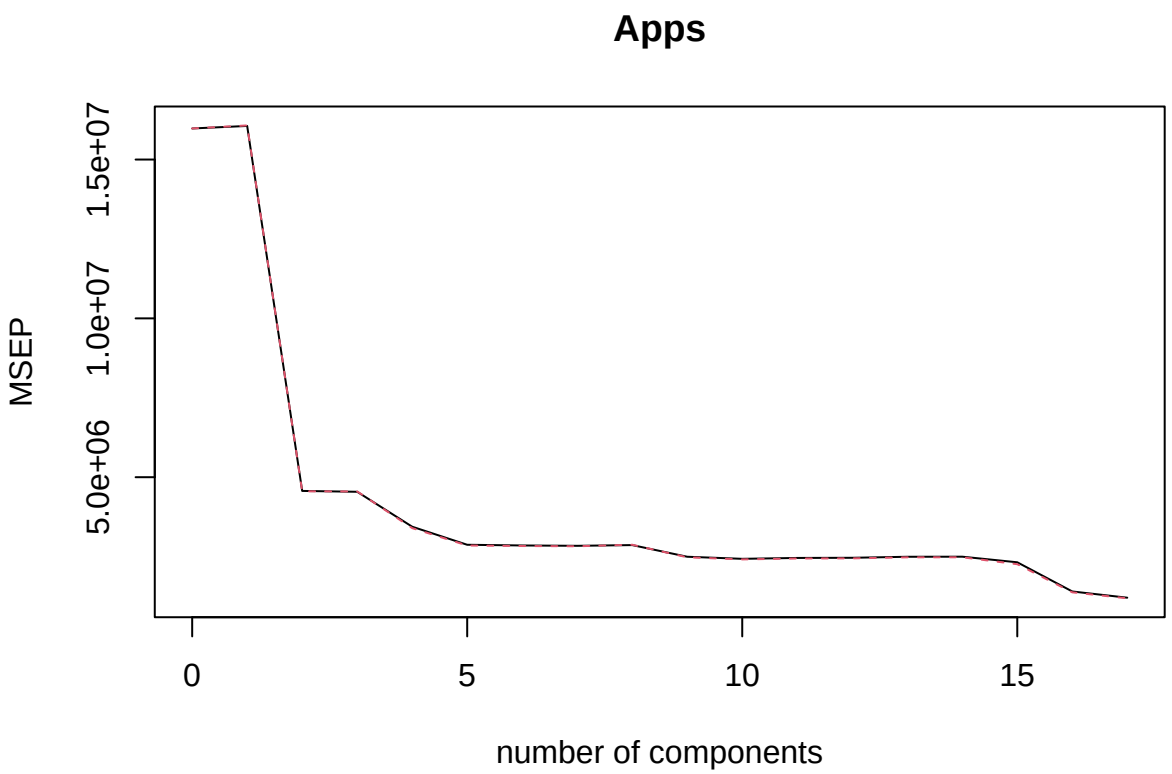
e.

```
# e. The PCR fit. M = 17 and test MSE is 1739432. This doesnt remove any variables.
library(pls)
```

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

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

```
set.seed(1321312)

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

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

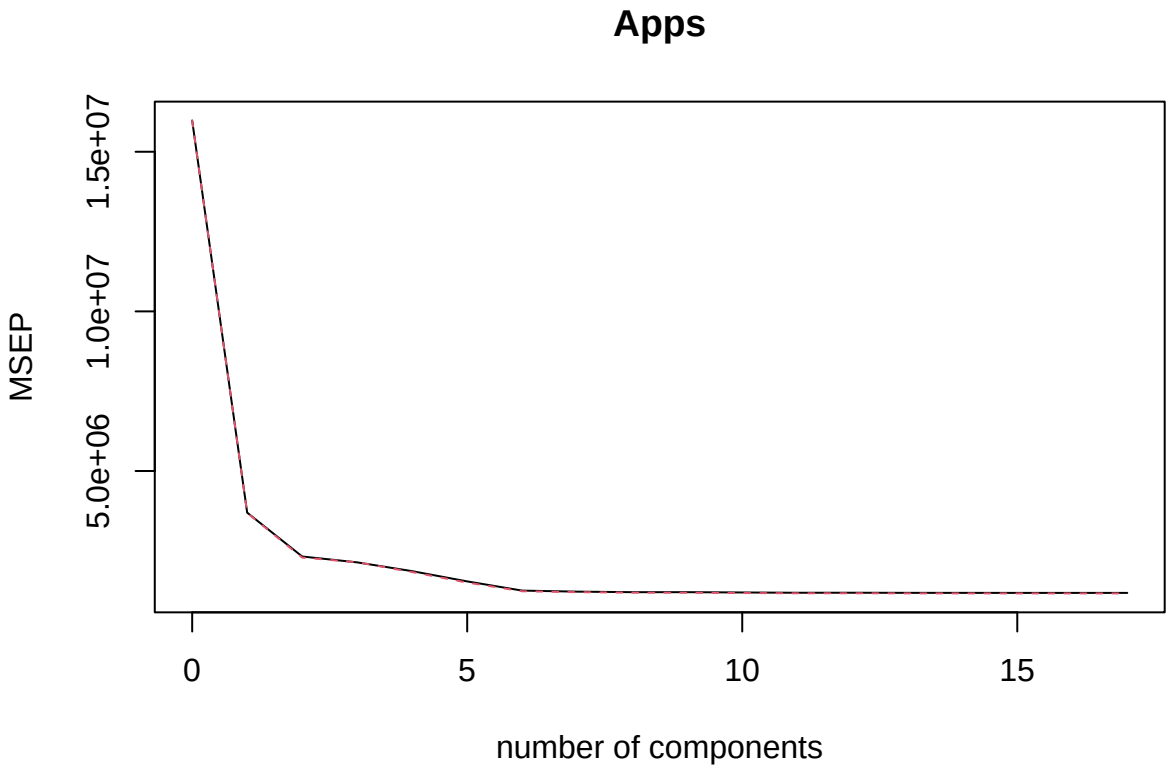



```
pcr.pred <- predict(pcr.fit, newdata = test, ncomp = 17)
pcr_mse <- mean((test$Apps - pcr.pred)^2)
print(pcr_mse)
```

```
## [1] 1739432
```

f.

```
# f. The PLS fit also chooses M=17 and the same test MSE as the PCR fit.
set.seed(123123)

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

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




```
pls.pred <- predict(pls.fit, newdata = test, ncomp = 17)
pls_mse  <- mean((test$Apps - pls.pred)^2)
print(pls_mse)
```

```
## [1] 1739432
```

g.

The only model that gives a noticeable improvement is with the Ridge regression model. LASSO provides a slightly better fit. PCR and PLS interestingly select M=17, which is essentially just the least squares fit. In the end, all of the fits are rather weak, so this model is likely not specified correctly.

Problem 11

a.

```
df <- Boston
# View(Boston)
```

```r
set.seed(2132)
train_idx <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train <- df[train_idx, ]
test  <- df[!train_idx, ]

X.train.matrix <- model.matrix(crim~., data=train)
# View(X.train.matrix)

# OLS benchmark
ls.fit <- lm(crim~X.train.matrix, data=train)
pred <- predict(ls.fit, newdata=test)
cat("Test MSE for OLS benchmark: ", mean((test$crim - pred)^2), "\n")
```

```
## Test MSE for OLS benchmark:  87.81298
```

```r
# Ridge
set.seed(12312)
X.train.matrix <- model.matrix(crim~.,data=train)[,-1]
Y.train <- train$crim

grid <- 10^seq(10,-2, length = 100)
cv.ridge.fit <- cv.glmnet(X.train.matrix, Y.train, standardize=TRUE,
                          nfolds=10, alpha=0)
best_lambda <- cv.ridge.fit$lambda.min
cat("Optimal lambda for ridge: ", best_lambda, "\n")
```

```
## Optimal lambda for ridge:  0.5620133
```

```r
X.test.matrix <- model.matrix(crim ~ ., data = test)[,-1]
preds <- predict(cv.ridge.fit, s = best_lambda, newx = X.test.matrix)
test_mse <- mean((test$crim - preds)^2)
cat("Test MSE for Ridge: ", test_mse, "\n")
```

```
## Test MSE for Ridge:  34.47066
```

```r
# LASSO
set.seed(12312)
X.train.matrix <- model.matrix(crim~.,data=train)[,-1]
Y.train <- train$crim

grid <- 10^seq(10,-2, length = 100)
cv.lasso.fit <- cv.glmnet(X.train.matrix, Y.train, standardize=TRUE,
                          nfolds=10, alpha=1)
best_lambda <- cv.lasso.fit$lambda.min
cat("Optimal lambda for lasso: ", best_lambda, "\n")
```

```
## Optimal lambda for lasso:  0.0588774
```

```r
X.test.matrix <- model.matrix(crim ~ ., data = test)[,-1]
preds <- predict(cv.lasso.fit, s = best_lambda, newx = X.test.matrix)
test_mse <- mean((test$crim - preds)^2)
cat("Test MSE for lasso: ", test_mse, "\n")
```

```
## Test MSE for lasso:  34.55797
```

b and c.

Based off of the results from (a), we will use the LASSO regression model. Though Ridge has a smaller test MSE, the difference between the LASSO (34.558) and Ridge (34.471) test MSE is small, so we will use LASSO regression for ease of inference and explicit variable selection. The final equation of the model is

$$\widehat{CrimeRate} = 8.9523 + 0.0367 \cdot zn - 0.0619 \cdot indus - 0.6588 \cdot chas - 6.2594 \cdot nox + 0.3790 \cdot rm - 0.7741 \cdot dis + 0.5363 \cdot rad - 0.2100 \cdot ptratio$$

The variables *age* and *tax* were eliminated from the regression. From this we can infer that these variables have little contribution to the model on crime rate.

```
Y <- df$crim
X.matrix <- model.matrix(crim~., data=df)[,-1]
# ?glmnet()
best.fit <-  glmnet(X.matrix, Y, lambda=0.0588774, alpha=1, standardize=TRUE)

predict(best.fit, type = "coefficients", s = 0.0588774)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  8.95227024
## zn           0.03673011
## indus       -0.06189160
## chas        -0.65875000
## nox         -6.25937785
## rm           0.37904406
## age          .
## dis         -0.77410881
## rad          0.53629386
## tax          .
## ptratio     -0.20997256
## lstat        0.13307172
## medv        -0.17412499
```