Tom Meagher
DS 502 Homework 3
October 25, 2016

**1.** Section 6.8, page 259, question 1

**a.**

Best subset selection has the smallest training RSS because forward and backward stepwise selection both follow a guided *search* over models, relying on the first predictor they pick. Since best subset consists of many more models (2^p), it will likely have the smallest training RSS because it does not rely on a path dependency.

**b.**

It depends. Best subset selection will likely have the smallest test RSS (and R^2) because there are substantially more models (2^p) than in either forward or backward stepwise selection. Best subset is very computationally expensive though. Forward and/or backward stepwise selection could result in a better fitting model than by chance without overfitting.

**c.**

**i.** True, the (k+1)-variable model contains all features in the k-variable model, and the best additional feature.

**ii.** True, the k-variable model contains all but one feature in the (k+1)-variable model, excluding the single feature that results in the smallest RSS gain.

**iii.** False, the predictors from forward and/or backward selection could be disjoint from each other.

**iv.** False, same as iii.

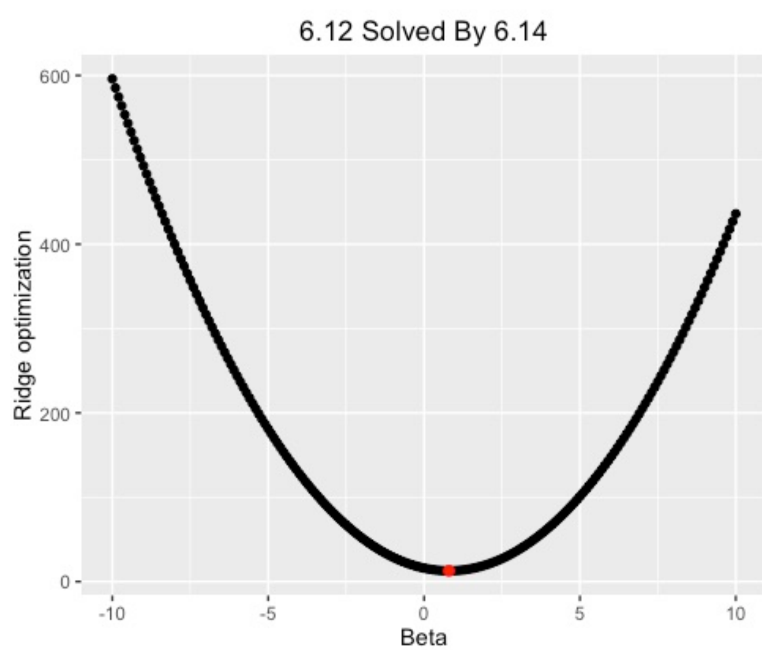
**v.** False, same as iii.

**2.** Section 6.8, page 261, question 6

**a.**

```
y <- 4
lambda <- 4
betas <- seq(-10, 10, 0.1)
func <- ((y - betas)^2) + (lambda * (betas^2))
est_beta <- y / (1 + lambda)
est_func <- ((y -
est_beta)^2) + (lambda *
(est_beta^2))

library(ggplot2)
ggplot() +
  geom_point(aes(x = betas, y
= func)) +
  geom_point(aes(x =
est_beta, y = est_func),
colour = "red", size = 2) +
  xlab("Beta") +
  ylab("Ridge optimization")
+
  labs(title = "6.12 Solved
By 6.14")
```



**b.**

```
y = 4
lambda = 4
betas = seq(-3, 3, 0.01)
func = ((y - betas)^2) + (lambda * abs(betas))
est_beta = y - lambda/2
est_func = ((y - est_beta)^2) + (lambda * abs(est_beta))

ggplot() +
  geom_point(aes(x = betas, y = func)) +
  geom_point(aes(x = est_beta, y = est_func), colour = "red", size = 2) +
  xlab("Beta") +
  ylab("Lasso optimization") +
  labs(title = "6.13 Solved By 6.15")
```


**3.** Section 6.8, page 262-263, question 8

**a.**

```
set.seed(1)
X <- rnorm(100)
epsilon <- rnorm(100)
```

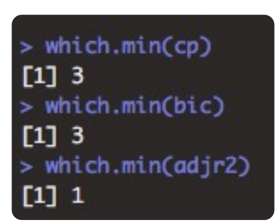
**b.**

```
beta_0 <- 1.5
beta_1 <- 20
beta_2 <- -1.5
beta_3 <- 0.1
Y <- beta_0 + beta_1 * X + beta_2 * X^2 + beta_3 * X^3 + epsilon
```

**c.**

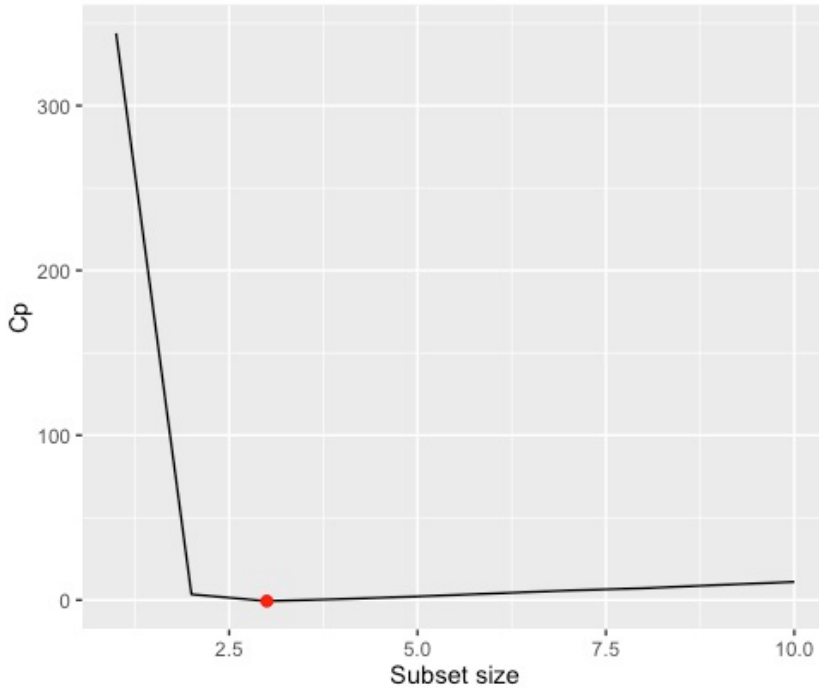
```
library(leaps)
data_frame = data.frame(y = Y, x = X)
regfit_full = regsubsets(y ~ poly(x, 10, raw = T), data =
data_frame, nvmax = 10)
regfit_summary = summary(regfit_full)

attach(regfit_summary)
which.min(cp)
which.min(bic)
which.min(adjr2)
```

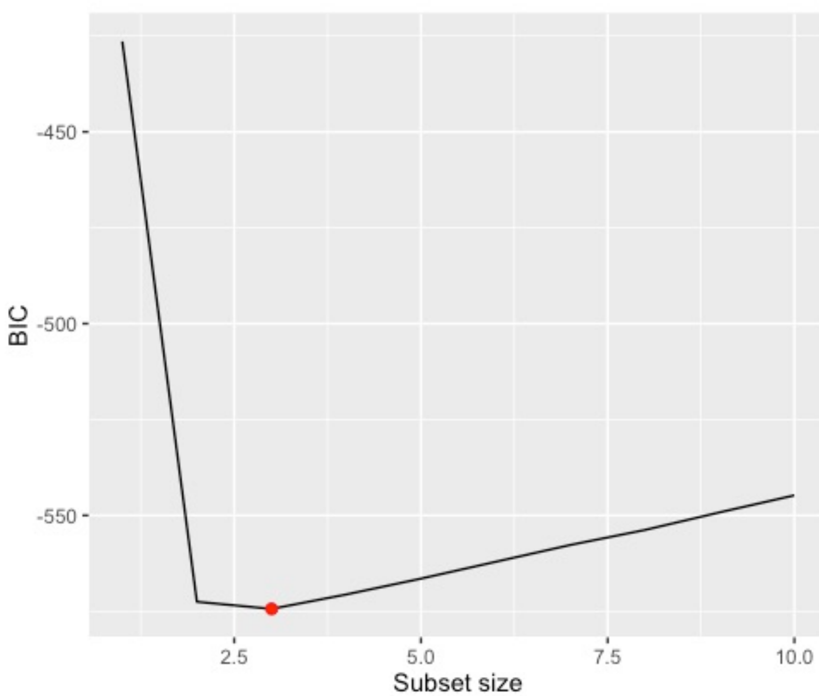


```
> which.min(cp)
[1] 3
> which.min(bic)
[1] 3
> which.min(adjr2)
[1] 1
```

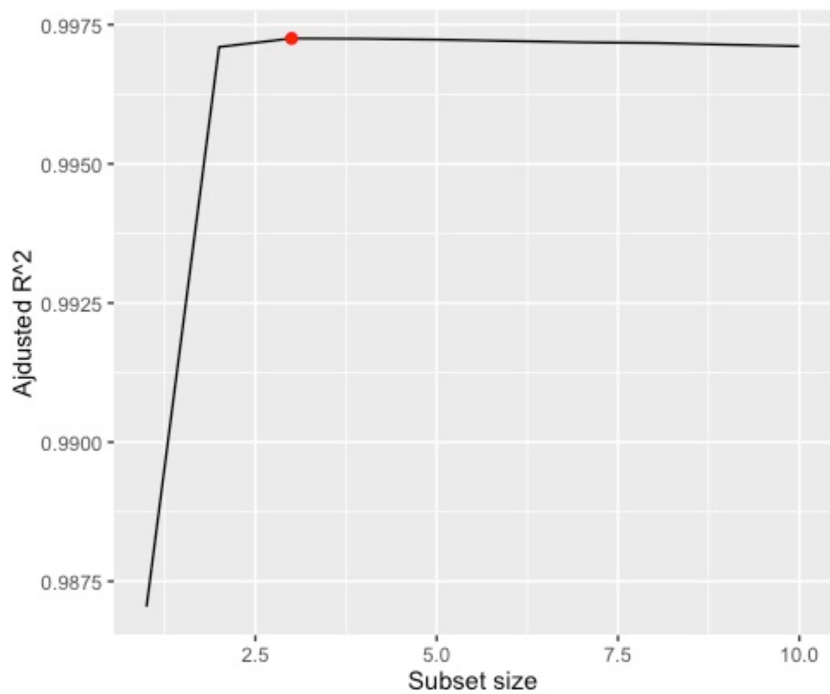
```
library(ggplot2)
ggplot() +
  geom_line(aes(x = seq(1, length(cp), 1), y = cp)) +
  geom_point(aes(x = 3, y = cp[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Cp")
```




```
ggplot() +
  geom_line(aes(x = seq(1, length(bic), 1), y = bic)) +
  geom_point(aes(x = 3, y = bic[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("BIC")
```



```
ggplot() +
  geom_line(aes(x = seq(1, length(adjr2), 1), y = adjr2)) +
  geom_point(aes(x = 3, y = adjr2[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Ajdusted R^2")
```




```
coefficients(regfit_full, id = 3)
```

```
> coefficients(regfit_full, id=3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)9
        1.579475231          20.105467476          -1.659286180           0.000900817
```

Based on the Cp, BIC, and adjusted R^2 values and plots, we see that model three performs the best. Interesting though is that which.min(adjr2) returns that model one performed the best, when the plot clearly shows it is three.

**d.**

```
forward_full = regsubsets(y ~ poly(x, 10, raw = T), data = data_frame, nvmax = 10,
method = "forward")
backward_full = regsubsets(y ~ poly(x, 10, raw = T), data = data_frame, nvmax = 10,
method = "backward")
forward_summary = summary(forward_full)
backward_summary = summary(backward_full)
```

```
which.min(forward_summary$cp)
which.min(forward_summary$bic)
which.min(forward_summary$adjr2)
```

```
> which.min(forward_summary$cp)
[1] 3
> which.min(forward_summary$bic)
[1] 3
> which.min(forward_summary$adjr2)
[1] 1
```

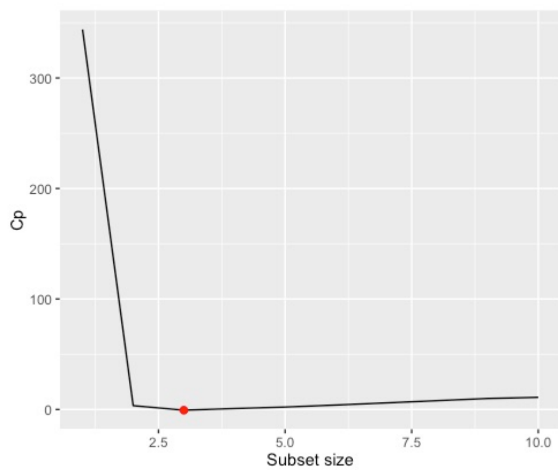
```
which.min(backward_summary$cp)
which.min(backward_summary$bic)
which.min(backward_summary$adjr2)
```

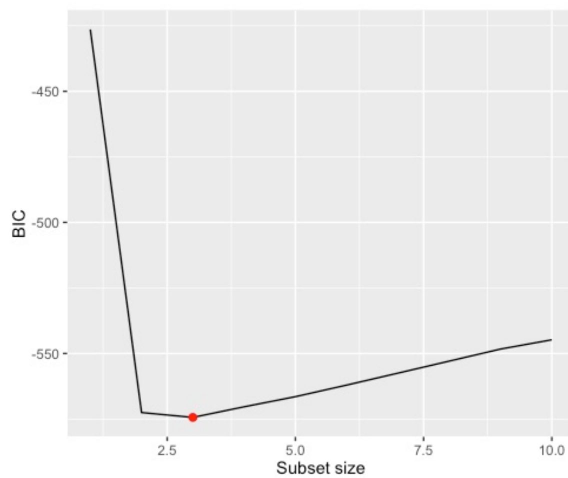
```
> which.min(backward_summary$cp)
[1] 3
> which.min(backward_summary$bic)
[1] 3
> which.min(backward_summary$adjr2)
[1] 1
```

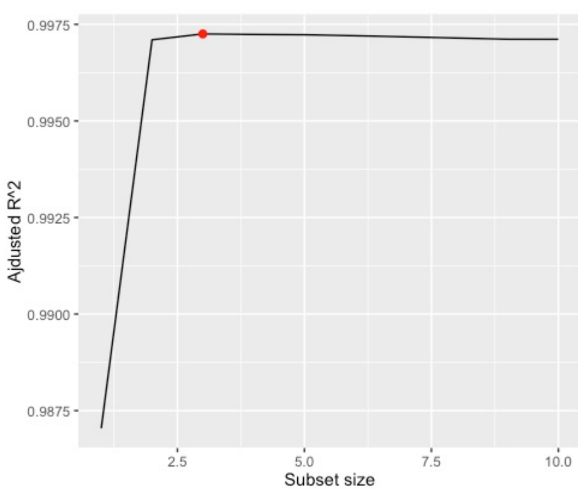
```
ggplot() +
  geom_line(aes(x = seq(1, length(forward_summary$cp), 1), y = forward_summary$cp)) +
  geom_point(aes(x = 3, y = forward_summary$cp[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Cp")

ggplot() +
  geom_line(aes(x = seq(1, length(forward_summary$bic), 1), y = forward_summary$bic))
+
  geom_point(aes(x = 3, y = forward_summary$bic[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("BIC")

ggplot() +
  geom_line(aes(x = seq(1, length(forward_summary$adjr2), 1), y =
forward_summary$adjr2)) +
  geom_point(aes(x = 3, y = forward_summary$adjr2[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Ajdusted R^2")
```



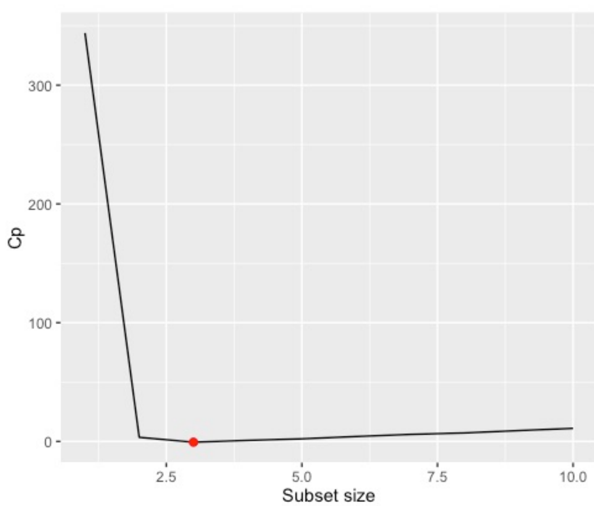



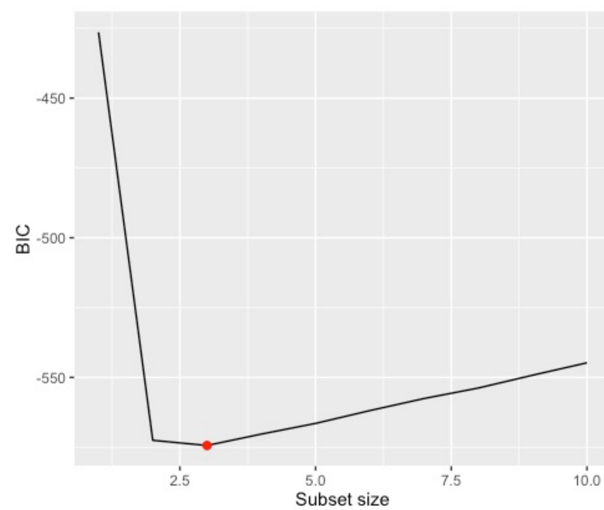



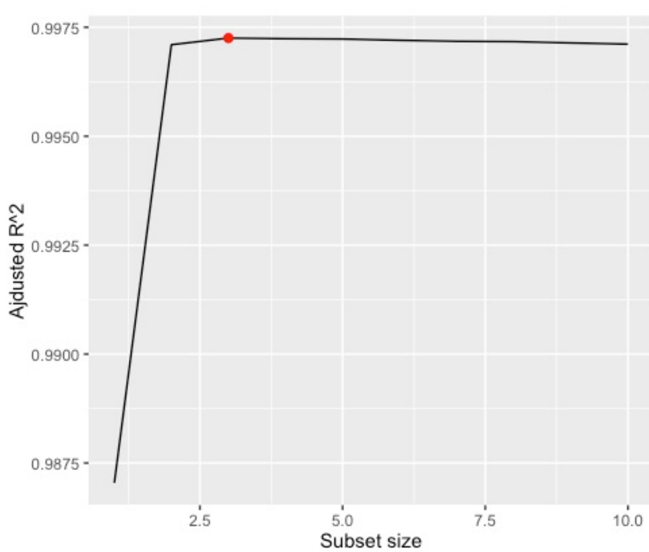
```
ggplot() +
  geom_line(aes(x = seq(1, length(backward_summary$cp), 1), y = backward_summary$cp))
+
  geom_point(aes(x = 3, y = backward_summary$cp[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Cp")

ggplot() +
  geom_line(aes(x = seq(1, length(backward_summary$bic), 1), y =
backward_summary$bic)) +
  geom_point(aes(x = 3, y = backward_summary$bic[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("BIC")

ggplot() +
  geom_line(aes(x = seq(1, length(backward_summary$adjr2), 1), y =
backward_summary$adjr2)) +
  geom_point(aes(x = 3, y = backward_summary$adjr2[3]), colour = "red", size = 2) +
  xlab("Subset size") +
  ylab("Ajdusted R^2")
```



```
coefficients(forward_full, id = 3)
coefficients(backward_full, id = 3)
coefficients(forward_full, id = 1)
coefficients(backward_full, id = 1)
```

```
> coefficients(forward_full, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)9
        1.579475231          20.105467476          -1.659286180           0.000900817
> coefficients(backward_full, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)9
        1.579475231          20.105467476          -1.659286180           0.000900817
> coefficients(forward_full, id = 1)
        (Intercept) poly(x, 10, raw = T)1
          0.2662558            20.0107755
> coefficients(backward_full, id = 1)
        (Intercept) poly(x, 10, raw = T)1
          0.2662558            20.0107755
```

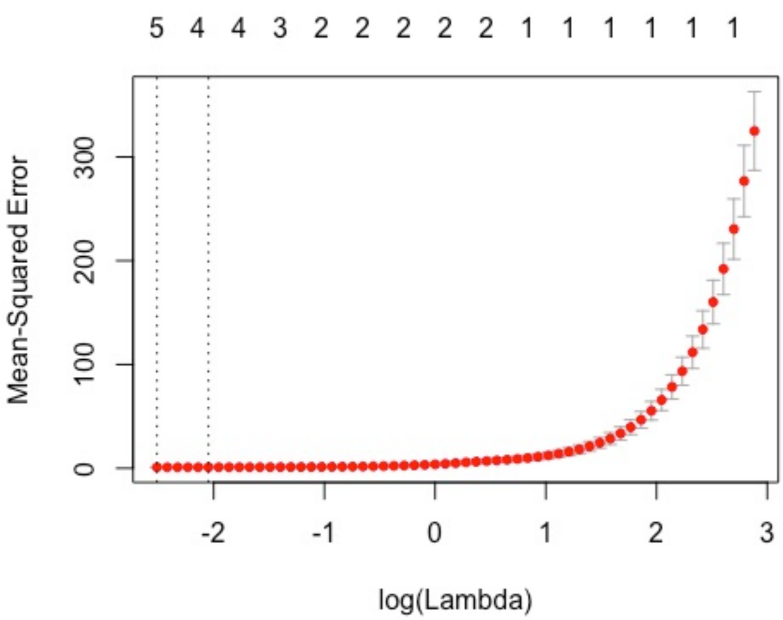
Based on the Cp, BIC, and adjusted R^2 values and plots for forward and backward selection, we see that model three performs the best. Noteworthy is that which.min(adjr2) for both forward and backward selection returns that model one performed the best, when the plot clearly shows it is three. The results are consistent with what was found in part c.

**e.**

```
library(glmnet)
model_matrix = model.matrix(y ~ poly(x, 10, raw = T), data = data_frame)[, -1]
lasso = cv.glmnet(model_matrix, Y, alpha = 1)
lambda_best = lasso$lambda.min
lambda_best

plot(lasso)
```

```
> best_lambda
[1] 0.08109982
```



```
model_best = glmnet(xmat, Y, alpha = 1)
predict(model_best, s = lambda_best, type = "coefficients")
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                                 1
(Intercept)           1.507517e+00
poly(x, 10, raw = T)1  2.001526e+01
poly(x, 10, raw = T)2 -1.550228e+00
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  .
poly(x, 10, raw = T)5  7.775514e-03
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  1.766705e-03
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  4.054021e-05
poly(x, 10, raw = T)10 .
```

The best Lasso model chose five coefficients over three.

**f.**

```
beta_7 <- 7
Y <- beta_0 + beta_7 * X^7 + epsilon
data_frame <- data.frame(y = Y, x = X)
regfit_full <- regsubsets(y ~ poly(x, 10, raw = T), data = data_frame, nvmax = 10)
regfit_summary <- summary(regfit_full)
```

```
which.min(regfit_summary$cp)
which.min(regfit_summary$bic)
which.min(regfit_summary$adjr2)
```

```
> which.min(regfit_summary$cp)
[1] 2
> which.min(regfit_summary$bic)
[1] 1
> which.min(regfit_summary$adjr2)
[1] 10
```

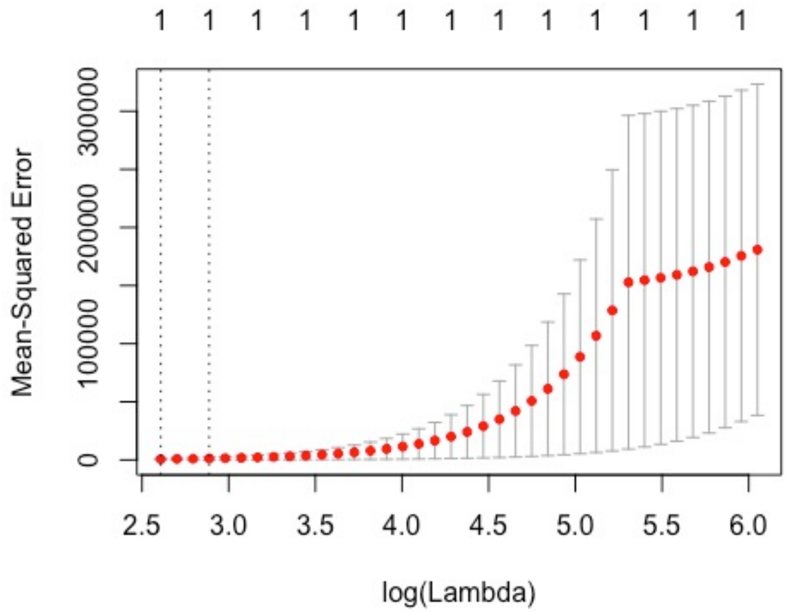
```
coefficients(regfit_full, id = 2)
coefficients(regfit_full, id = 1)
coefficients(regfit_full, id = 10)
```

```
> coefficients(regfit_full, id = 2)
          (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
            1.5704904            -0.1417084             7.0015552
> coefficients(regfit_full, id = 1)
          (Intercept) poly(x, 10, raw = T)7
              1.45894               7.00077
> coefficients(regfit_full, id = 10)
           (Intercept)  poly(x, 10, raw = T)1  poly(x, 10, raw = T)2  poly(x, 10, raw = T)3
            1.67282867             0.51409233            -1.13146007            -0.93113515
 poly(x, 10, raw = T)4  poly(x, 10, raw = T)5  poly(x, 10, raw = T)6  poly(x, 10, raw = T)7
            1.90382807             0.55109577            -1.26499408             6.84430680
 poly(x, 10, raw = T)8  poly(x, 10, raw = T)9 poly(x, 10, raw = T)10
            0.31986888             0.01627747            -0.02690171
```

```
xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data_frame)[, -1]
lasso = cv.glmnet(xmat, Y, alpha = 1)
lambda_best = lasso$lambda.min
lambda_best
```

```
> lambda_best
[1] 13.57478
```

```
plot(lasso)
```




```
model_best = glmnet(xmat, Y, alpha = 1)
predict(model_best, s = lambda_best, type = "coefficients")
```

```
> predict(model_best, s = lambda_best, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)            2.404188
poly(x, 10, raw = T)1  .
poly(x, 10, raw = T)2  .
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  .
poly(x, 10, raw = T)5  .
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  6.776797
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
```

The best Lasso model chose seven coefficients.

**4.** Section 6.8, page 263, question 9

**a.**

```
library(ISLR)
set.seed(11)
train_size = dim(College)[1] / 2
train = sample(1:dim(College)[1], train_size)
test = -train
college_train = College[train,]
college_test = College[test,]
```

**b.**

```
lm_fit = lm(Apps ~ ., data = college_train)
lm_pred = predict(lm_fit, college_test)
mean((college_test[, "Apps"] - lm_pred)^2)
```

```
> mean((college_test[, "Apps"] - lm_pred)^2)
[1] 1538442
```

The test RSS is 1,538,442.

**c.**

```
library(glmnet)
matrix_train = model.matrix(Apps ~ ., data = college_train)
matrix_test = model.matrix(Apps ~ ., data = college_test)
grid = 10 ^ (seq(4, -2, length=100))
model_ridge <- cv.glmnet(matrix_train, college_train[, "Apps"], alpha = 0, lambda =
grid, thresh = 1e-12)
lambda_best <- model_ridge$lambda.min
lambda_best
```

```
> lambda_best
[1] 8.111308
```

```
ridge_regression = predict(model_ridge, newx = matrix_test, s = lambda_best)
mean((college_test[, "Apps"] - ridge_regression)^2)
```

```
> mean((college_test[, "Apps"] - ridge_regression)^2)
[1] 1568103
```

The test RSS is 1,568,103.

**d.**

```
model_lasso = cv.glmnet(matrix_train, college_train[, "Apps"], alpha = 1, lambda =
grid, thresh = 1e-12)
lambda_best = model_lasso$lambda.min
lambda_best
```

```
> lambda_best
[1] 21.54435
```

```
lasso_prediction = predict(model_lasso, newx = matrix_test, s = lambda_best)
mean((college_test[, "Apps"] - lasso_prediction)^2)
```

```
> mean((college_test[, "Apps"] - lasso_prediction)^2)
[1] 1635280
```

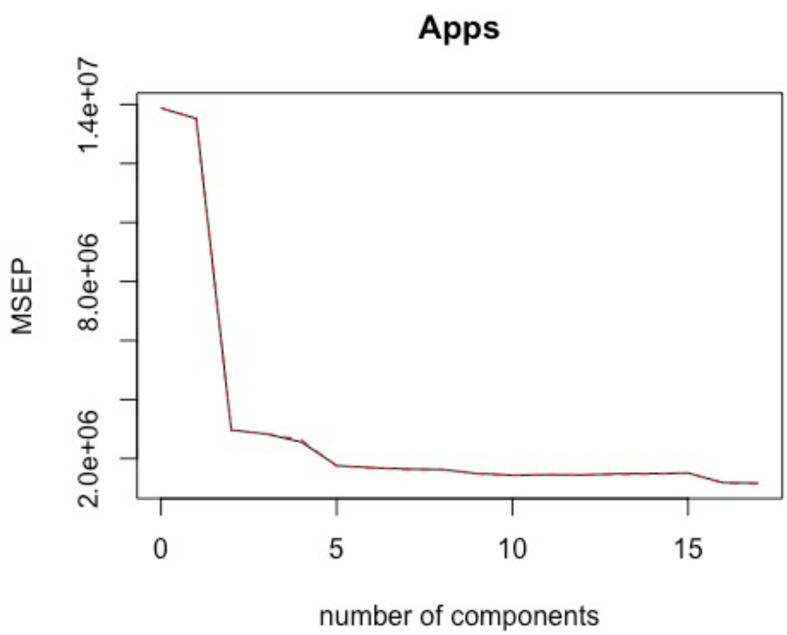
The test RSS is 1,635,280.

```
sum(coef(model_lasso)[,1] == 0)
names(coef(model_lasso)[, 1][coef(model_lasso)[, 1] == 0])
```

```
> names(coef(model_lasso)[, 1][coef(model_lasso)[, 1] == 0])
 [1] "(Intercept)" "Enroll"      "Top25perc"   "P.Undergrad" "Outstate"    "Room.Board"  "Books"
 [8] "Personal"    "PhD"         "Terminal"    "S.F.Ratio"   "perc.alumni" "Grad.Rate"
```

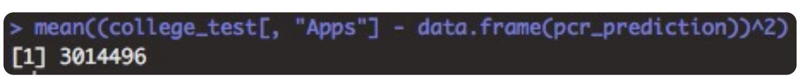
Enroll, Top25perc, P.Undergrad, Outstate, Room.Board, Books, Personal, PhD, Terminal, S.F.Ratio, perc.alumni, and Grad.Rate are all non-zero. So there are 13 non-zero coefficients.

**e.**
```
library(pls)
pcr_fit = pcr(Apps ~ ., data = college_train, scale = T, validation = "CV")
validationplot(pcr_fit, val.type = "MSEP")
```




```
pcr_prediction = predict(pcr_fit, college_test, ncomp = 10)
mean((college_test[, "Apps"] - data.frame(pcr_prediction))^2)
```

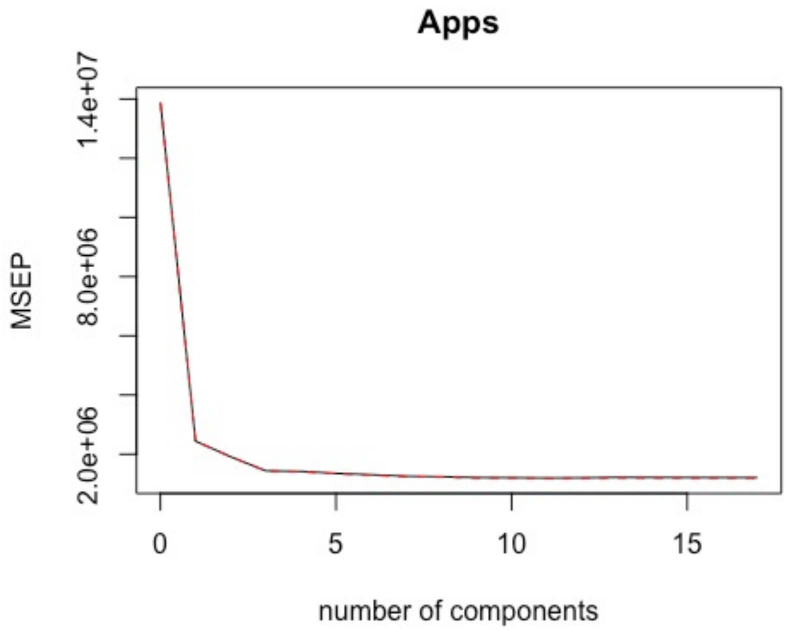
The test RSS is 3,014,496.



```
> mean((college_test[, "Apps"] - data.frame(pcr_prediction))^2)
[1] 3014496
```

**f.**
```
pls_fit = plsr(Apps ~ ., data = college_train, scale = T, validation = "CV")
validationplot(pls_fit, val.type = "MSEP")
```

(see next page)




```
pls_prediction = predict(pls_fit, college_test, ncomp = 10)
mean((college_test[, "Apps"] - data.frame(pls_prediction))^2)
```

The test RSS is 1,508,987.

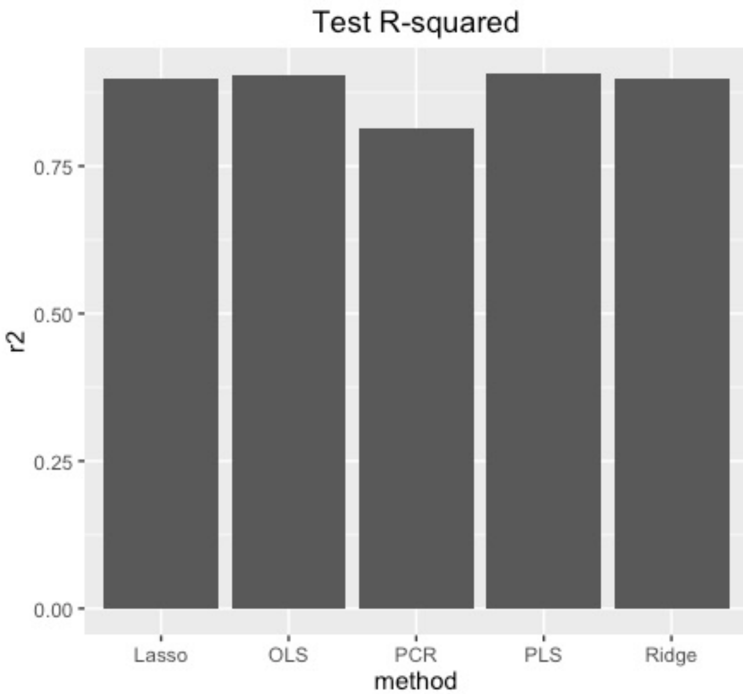
```
> mean((college_test[, "Apps"] - data.frame(pls_prediction))^2)
[1] 1508987
```

**g.**

```
test_avg = mean(college_test[, "Apps"])
mean_college_test = mean((college_test[, "Apps"] - test_avg)^2)

lm_r2 = 1 - mean((college_test[, "Apps"] - lm_pred)^2) / mean_college_test
ridge_r2 = 1 - mean((college_test[, "Apps"] - ridge_regression)^2) /
mean_college_test
lasso_r2 = 1 - mean((college_test[, "Apps"] - lasso_pred)^2) / mean_college_test
pcr_r2 = 1 - mean((college_test[, "Apps"] - data.frame(pcr_pred))^2) /
mean_college_test
pls_r2 = 1 - mean((college_test[, "Apps"] - data.frame(pls_pred))^2) /
mean_college_test

results <- data.frame(method = c("OLS", "Ridge", "Lasso", "PCR", "PLS"), r2 =
c(lm_r2, ridge_r2, lasso_r2, pcr_r2, pls_r2))
ggplot(results, aes(method, r2)) +
  geom_bar(stat = "identity") +
  labs(title = "Test R-squared")
```




OLS and PLS performed the best, followed by Lasso and Ridge, then PCR. All of the approaches do a good job at predicting the number of college applications received because their R^2 values mean that at least 75% of number of application received can be explained by the variables in the models. It doesn't appear there is much of a difference between the five approaches, but it depends on the domain. Maybe a 5% decrease in accuracy is extremely expensives, whereas in college application it might be okay.