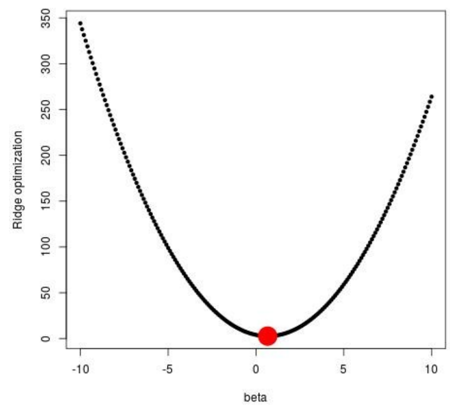
6)
a)
For p=1, (6.12) takes the form $(y-\beta)^2+\lambda\beta^2$.
We plot this function for y=2, λ=2.

```
> y = 2
> lambda = 2
> 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
> jpeg("6a.jpg")
> plot(betas, func, pch = 20, xlab = "beta", ylab = "Ridge optimization")
> points(est.beta, est.func, col = "red", pch = 4, lwd = 5, cex = est.beta)
> dev.off()
```

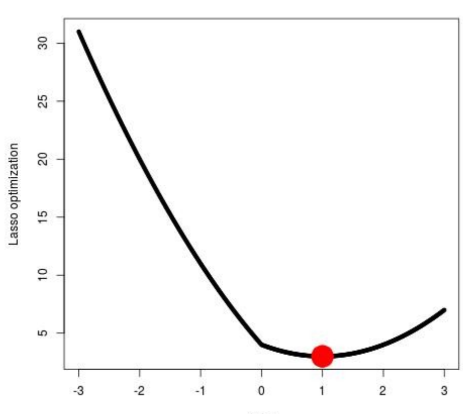
The big dot shows that function is minimized at $\beta=y/(1+\lambda)$



b)
For p=1, (6.13) takes the form $(y-\beta)2+\lambda|\beta|$.
We plot this function for y=2,λ=2.

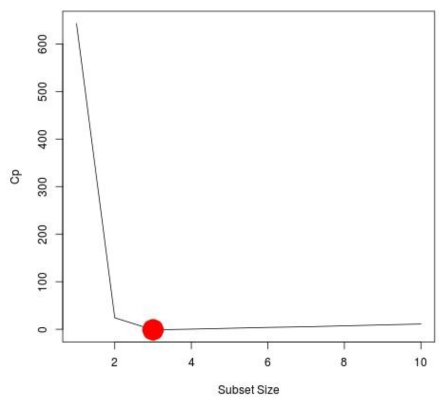
```
> y = 2
> lambda = 2
> betas = seq(-3, 3, 0.01)
> func = (y - betas)^2 + lambda * abs(betas)
> est.beta = y - lambda/2
> est.func = (y - est.beta)^2 + lambda * abs(est.beta)
> jpeg("6b.jpg")
> plot(betas, func, pch = 20, xlab = "beta", ylab = "Lasso optimization")
> points(est.beta, est.func, col = "red", pch = 8, lwd = 20, cex = est.beta)
> dev.off()
```

The big dot shows that function is indeed minimized at $\beta=y-\lambda/2$.

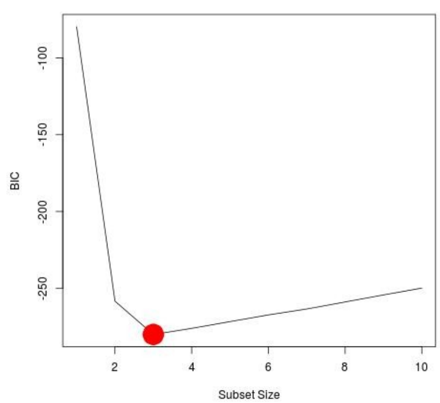


8)
a)
```
> set.seed(1)
> X = rnorm(100)
> eps = rnorm(100)
```

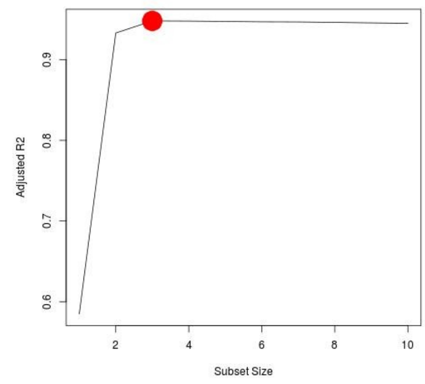
b)
We are selecting β0=3, β1=2, β2=−3 and β3=0.3.
```
> beta0 = 3
> beta1 = 2
> beta2 = -3
> beta3 = 0.3
> Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps

install.packages("leaps")
> library(leaps)
> data.full = data.frame(y = Y, x = X)
> mod.full = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10)
> mod.summary = summary(mod.full)
>
> # Best cp, BIC and adjr2
> which.min(mod.summary$cp)
[1] 3
> which.min(mod.summary$bic)
[1] 3
> which.max(mod.summary$adjr2)
[1] 3
> # Plot cp, BIC and adjr2
> jpeg("8a.jpg")
> plot(mod.summary$cp, xlab = "Subset Size", ylab
= "Cp", pch = 20, type = "l")
> points(3, mod.summary$cp[3], pch = 8, col =
"red", lwd = 20)
> dev.off()
```




```
> jpeg("8a1.jpg")
> plot(mod.summary$bic, xlab = "Subset Size", ylab
= "BIC", pch = 20, type = "l")
> points(3, mod.summary$bic[3], pch = 8, col =
"red", lwd = 20)
> dev.off()
```



```
> jpeg("8a2.jpg")
> plot(mod.summary$adjr2, xlab = "Subset Size", ylab =
"Adjusted R2", pch = 20, type = "l")
> points(3, mod.summary$adjr2[3], pch = 8, col = "red",
lwd = 20)
> dev.off()
```



We find that with Cp, BIC and Adjusted R2 criteria, 3, 3, and 3 variable models are respectively picked.

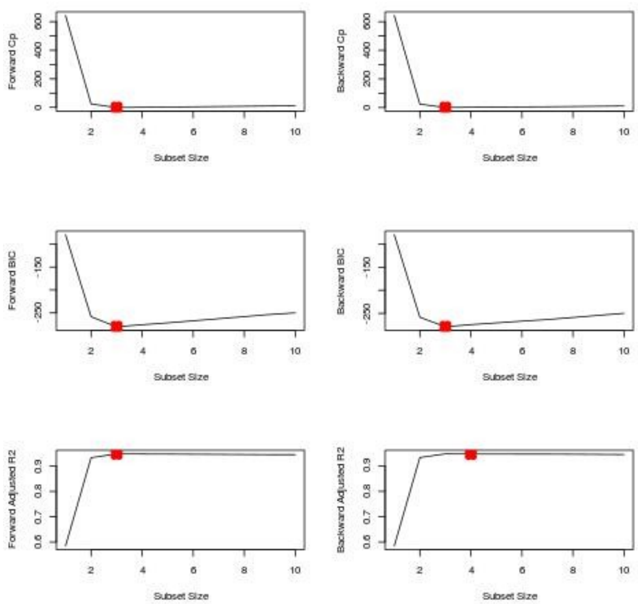
```
> coefficients(mod.full, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
         3.07627412            2.35623596          -3.16514887
poly(x, 10, raw = T)7
         0.01046843
```

All statistics pick X^7 over X^3. The remaining coefficients are quite close to β s.

d)
We fit forward and backward stepwise models to the data.

```
> mod.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10, method = "forward")
> mod.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10, method = "backward")
> fwd.summary = summary(mod.fwd)
> bwd.summary = summary(mod.bwd)

> which.min(fwd.summary$cp)
[1] 3
> which.min(bwd.summary$cp)
[1] 3
> which.min(fwd.summary$bic)
[1] 3
> which.min(bwd.summary$bic)
[1] 3
> which.max(fwd.summary$adjr2)
[1] 3
> which.max(bwd.summary$adjr2)
[1] 3

> jpeg("6d.jpg")
> par(mfrow = c(3, 2))
> plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward Cp", pch = 20, type = "l")
> points(3, fwd.summary$cp[3], pch = 4, col = "red", lwd = 7)
> plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward Cp", pch = 20, type = "l")
> points(3, bwd.summary$cp[3], pch = 4, col = "red", lwd = 7)
> plot(fwd.summary$bic, xlab = "Subset Size", ylab = "Forward BIC", pch = 20,
+     type = "l")
> points(3, fwd.summary$bic[3], pch = 4, col = "red", lwd = 7)
```

```
> plot(bwd.summary$bic, xlab = "Subset Size", ylab = "Backward BIC", pch = 20,
+     type = "l")
> points(3, bwd.summary$bic[3], pch = 4, col = "red", lwd = 7)
> plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R2",
+     pch = 20, type = "l")
> points(3, fwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
> plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R2",
+     pch = 20, type = "l")
> points(4, bwd.summary$adjr2[4], pch = 4, col = "red", lwd = 7)
> dev.off()
```



We see that all statistics pick 3 variable models except backward stepwise with adjusted R2.
Here are the coefficients:

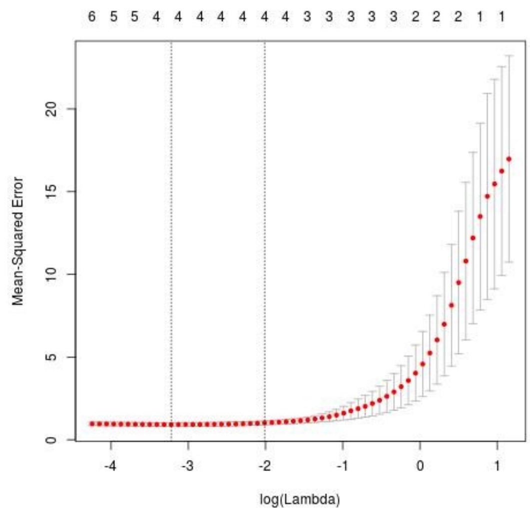
```
> coefficients(mod.fwd, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
         3.07627412          2.35623596         -3.16514887
poly(x, 10, raw = T)7
         0.01046843

> coefficients(mod.bwd, id = 3)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
        3.078881355          2.419817953         -3.177235617
poly(x, 10, raw = T)9
        0.001870457

> coefficients(mod.fwd, id = 4)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
        3.112358625          2.369858879         -3.275726574
poly(x, 10, raw = T)4 poly(x, 10, raw = T)7
        0.027673638          0.009997134
```

Here forward stepwise picks X^7 over X^3. Backward stepwise with 3 variables picks X^9 while backward stepwise with 4 variables picks X^4 and X^7. All other coefficients are close to β s.

c)

Training Lasso on the data

```
> library(glmnet)
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]
> mod.lasso = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = mod.lasso$lambda.min
> best.lambda
[1] 0.03991416

> jpeg("8e.jpg")
> plot(mod.lasso)
> dev.off()
```




```
> # Next fit the model on entire data using best lambda
> best.model = glmnet(xmat, Y, alpha = 1)
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                          1
(Intercept)           3.0398151056
poly(x, 10, raw = T)1   2.2303371338
poly(x, 10, raw = T)2  -3.1033192679
poly(x, 10, raw = T)3   .
poly(x, 10, raw = T)4   .
poly(x, 10, raw = T)5   0.0498410763
poly(x, 10, raw = T)6   .
poly(x, 10, raw = T)7   0.0008068431
poly(x, 10, raw = T)8   .
poly(x, 10, raw = T)9   .
poly(x, 10, raw = T)10  .
```

Lasso also picks X^5 over X^3. It also picks X^7 with negligible coefficient.

f)

Create new Y with different β7=7.

```
> beta7 = 7
> Y = beta0 + beta7 * X^7 + eps
> # Predict using regsubsets
> data.full = data.frame(y = Y, x = X)
> mod.full = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10)
> mod.summary = summary(mod.full)

> # Best cp, BIC and adjr2
> which.min(mod.summary$cp)
[1] 2
```

```
> which.min(mod.summary$bic)
[1] 1
> which.max(mod.summary$adjr2)
[1] 4
> coefficients(mod.full, id = 1)
        (Intercept) poly(x, 10, raw = T)7
           2.95894              7.00077
> coefficients(mod.full, id = 2)
        (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
          3.0704904          -0.1417084            7.0015552
> coefficients(mod.full, id = 4)
        (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
          3.0762524           0.2914016           -0.1617671
poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
         -0.2526527            7.0091338
```

We see that BIC picks the most accurate 1-variable model with matching coefficients. Other criteria pick additional variables.

```
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]
> mod.lasso = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = mod.lasso$lambda.min
> best.lambda
[1] 13.57478

> best.model = glmnet(xmat, Y, alpha = 1)
> predict(best.model, s = best.lambda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                           1
(Intercept)           3.904188
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 .
```

Lasso also picks the best 1-variable model but intercet is quite off (3.8 vs 3).

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)
[1] 1538442
```

Test RSS is 1538442

c)

Pick λ using College.train and report error on College.test

```
> train.mat = model.matrix(Apps~., data=College.train)
> test.mat = model.matrix(Apps~., data=College.test)
> grid = 10 ^ seq(4, -2, length=100)
> mod.ridge = cv.glmnet(train.mat, College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)
> lambda.best = mod.ridge$lambda.min
> lambda.best
[1] 18.73817

> ridge.pred = predict(mod.ridge, newx=test.mat, s=lambda.best)
> mean((College.test[, "Apps"] - ridge.pred)^2)
[1] 1608859
```

Test RSS is slightly higher that OLS, 1608859.

d)

Pick λ using College.train and report error on College.test

```
> mod.lasso = cv.glmnet(train.mat, College.train[, "Apps"], alpha=1, lambda=grid, thresh=1e-12)
> lambda.best = mod.lasso$lambda.min
> lambda.best
[1] 21.54435

> lasso.pred = predict(mod.lasso, newx=test.mat, s=lambda.best)
> mean((College.test[, "Apps"] - lasso.pred)^2)
[1] 1635280
```

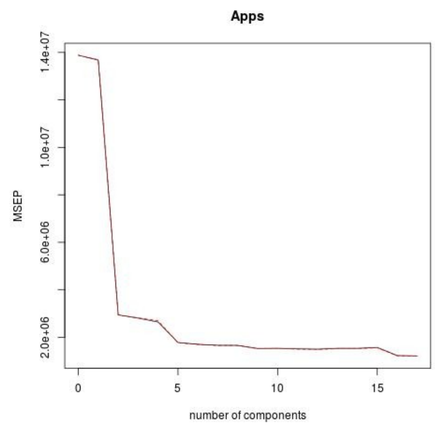
Again, Test RSS is slightly higher that OLS, 1635280.

The coefficients look like

```
> mod.lasso = glmnet(model.matrix(Apps~., data=College), College[, "Apps"], alpha=1)
> predict(mod.lasso, s=lambda.best, type="coefficients")
19 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -6.038452e+02
(Intercept)  .
PrivateYes  -4.235413e+02
```

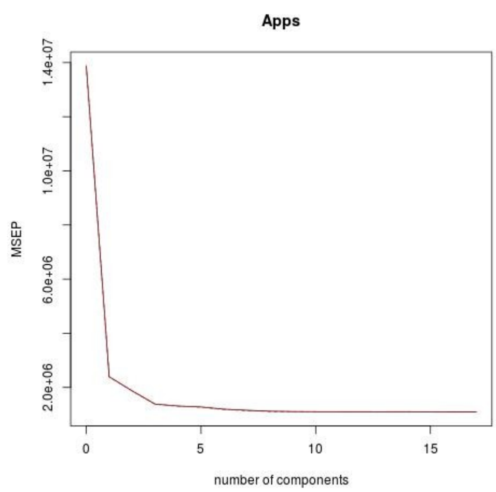
Accept 1.455236e+00
Enroll -2.003696e-01
Top10perc 3.367640e+01
Top25perc -2.403036e+00
F.Undergrad .
P.Undergrad 2.086035e-02
Outstate -5.781855e-02
Room.Board 1.246462e-01
Books .
Personal 1.832912e-05
PhD -5.601313e+00
Terminal -3.313824e+00
S.F.Ratio 4.478684e+00
perc.alumni -9.796600e-01
Expend 6.967693e-02
Grad.Rate 5.159652e+00

e)
```
> jpeg("9e.jpg")
> pcr.fit = pcr(Apps~., data=College.train, scale=T,
validation="CV")
> validationplot(pcr.fit, val.type="MSEP")
> dev.off()

> pcr.pred = predict(pcr.fit, College.test, ncomp=10)
> mean((College.test[, "Apps"] - data.frame(pcr.pred))^2)
[1] 3014496
```



Test RSS for PCR is about 3014496.

f)
```
> jpeg("9f.jpg")
> pls.fit = plsr(Apps~., data=College.train, scale=T,
validation="CV")
> validationplot(pls.fit, val.type="MSEP")
> dev.off()

> pls.pred = predict(pls.fit, College.test, ncomp=10)
> mean((College.test[, "Apps"] -
data.frame(pls.pred))^2)
[1] 1508987
```

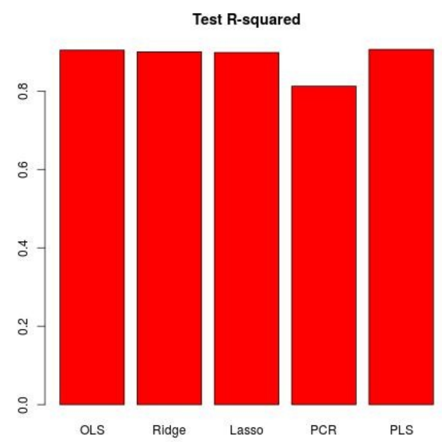


Test RSS for PLS is about 1508987.

g)
Results for OLS, Lasso, Ridge are comparable. Lasso reduces the F. Undergrad and Books variables to zero and shrinks coefficients of other variables. Here are the test R^2 for all models.

```
> test.avg = mean(College.test[, "Apps"])
```

```
> lm.test.r2 = 1 - mean((College.test[, "Apps"] - lm.pred)^2) /mean((College.test[, "Apps"] -
test.avg)^2)
> ridge.test.r2 = 1 - mean((College.test[, "Apps"] - ridge.pred)^2) /mean((College.test[, "Apps"] -
test.avg)^2)
> lasso.test.r2 = 1 - mean((College.test[, "Apps"] - lasso.pred)^2) /mean((College.test[, "Apps"] -
test.avg)^2)
> pcr.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pcr.pred))^2) /mean((College.test[,
"Apps"] - test.avg)^2)
> pls.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pls.pred))^2) /mean((College.test[,
"Apps"] – test.avg)^2)

> jpeg("9g.jpg")
> barplot(c(lm.test.r2, ridge.test.r2, lasso.test.r2, pcr.test.r2, pls.test.r2), col="red", names.arg=c("OLS",
"Ridge", "Lasso", "PCR", "PLS"), main="Test R-squared")
> dev.off()
```



The plot shows that test R^2 for all models except PCR are around 0.9, with PLS having slightly higher test R^2 than others. PCR has a smaller test R^2 of less than 0.8. All models except PCR predict college applications with high accuracy.

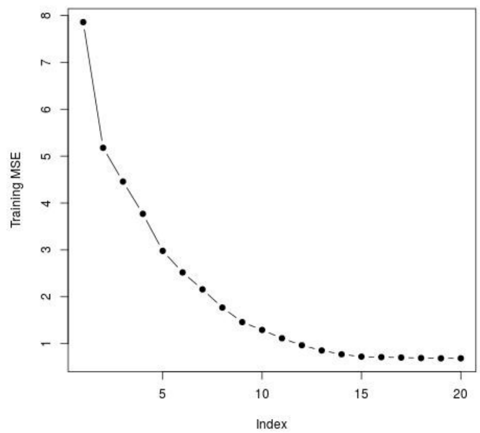
10)
a)

```
> set.seed(1)
> p = 20
> n = 1000
> x = matrix(rnorm(n * p), n, p)
> B = rnorm(p)
> B[3] = 0
> B[4] = 0
> B[9] = 0
> B[19] = 0
> B[10] = 0
> eps = rnorm(p)
> y = x %*% B + eps
```

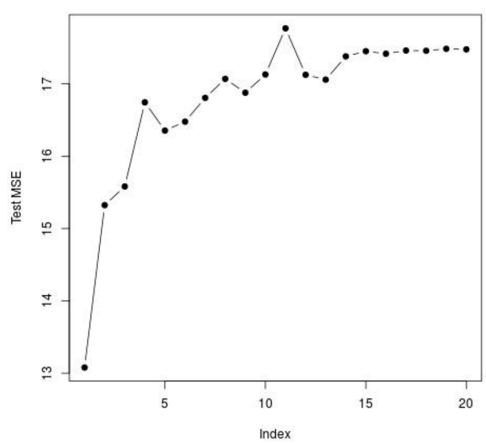
b)

```
> train = sample(seq(1000), 100, replace = FALSE)
> y.train = y[train, ]
> y.test = y[-train, ]
> x.train = x[train, ]
> x.test = x[-train, ]
```

c)
```
> library(leaps)
> regfit.full = regsubsets(y ~ ., data = data.frame(x = x.train, y = y.train),
+     nvmax = p)
> val.errors = rep(NA, p)
> x_cols = colnames(x, do.NULL = FALSE, prefix =
"x.")
> for (i in 1:p) {
+     coefi = coef(regfit.full, id = i)
+     pred = as.matrix(x.train[, x_cols %in%
names(coefi)]) %*% coefi[names(coefi) %in%
+         x_cols]
+     val.errors[i] = mean((y.train - pred)^2)
+ }

> jpeg("9c.jpg")
> plot(val.errors, ylab = "Training MSE", pch = 19,
type = "b")
> dev.off()
```



d)
```
> val.errors = rep(NA, p)
> for (i in 1:p) {
+     coefi = coef(regfit.full, id = i)
+     pred = as.matrix(x.test[, x_cols %in%
names(coefi)]) %*% coefi[names(coefi) %in%
+         x_cols]
+     val.errors[i] = mean((y.test - pred)^2)
+ }

> jpeg("10d.jpg")
> plot(val.errors, ylab = "Test MSE", pch = 19, type =
"b")
> dev.off()
```



c)
```
> which.min(val.errors)
[1] 1
```

16 parameter model has the smallest test MSE.

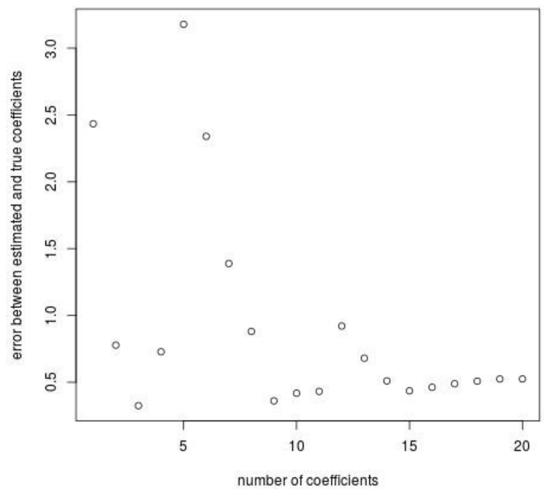
d)
```
> coef(regfit.full, id = 16)
(Intercept)         x.1         x.2         x.3         x.5         x.6 
-0.04912916  0.22060979  0.26843344  0.10422113  0.77976400 -0.40848211 
        x.7         x.8        x.11        x.12        x.13        x.14 
-1.39038502  0.70929994  0.71235190  0.63048827 -0.51885931 -0.74485348 
       x.15        x.16        x.17        x.18        x.20 
```

-0.65263105 -0.40271951 0.30008836 1.68412537 -0.95402290

Caught all but one zeroed out coefficient at x.19.

d)

```
> val.errors = rep(NA, p)
> a = rep(NA, p)
> b = rep(NA, p)
> for (i in 1:p) {
+     coefi = coef(regfit.full, id = i)
+     a[i] = length(coefi) - 1
+     b[i] = sqrt(sum((B[x_cols %in%
names(coefi)] - coefi[names(coefi) %in%
x_cols])^2) +
+         sum(B[!(x_cols %in% names(coefi))])^2)
+ }

> jpeg("10g.jpg")
> plot(x = a, y = b, xlab = "number of
coefficients", ylab = "error between estimated
and true coefficients")
> dev.off()

> which.min(b)
[1] 3
```



Model with 9 coefficients (10 with intercept) minimizes the error between the estimated and true coefficients. Test error is minimized with 16 parameter model. A better fit of true coefficients as measured here doesn't mean the model will have a lower test MSE.