HW2

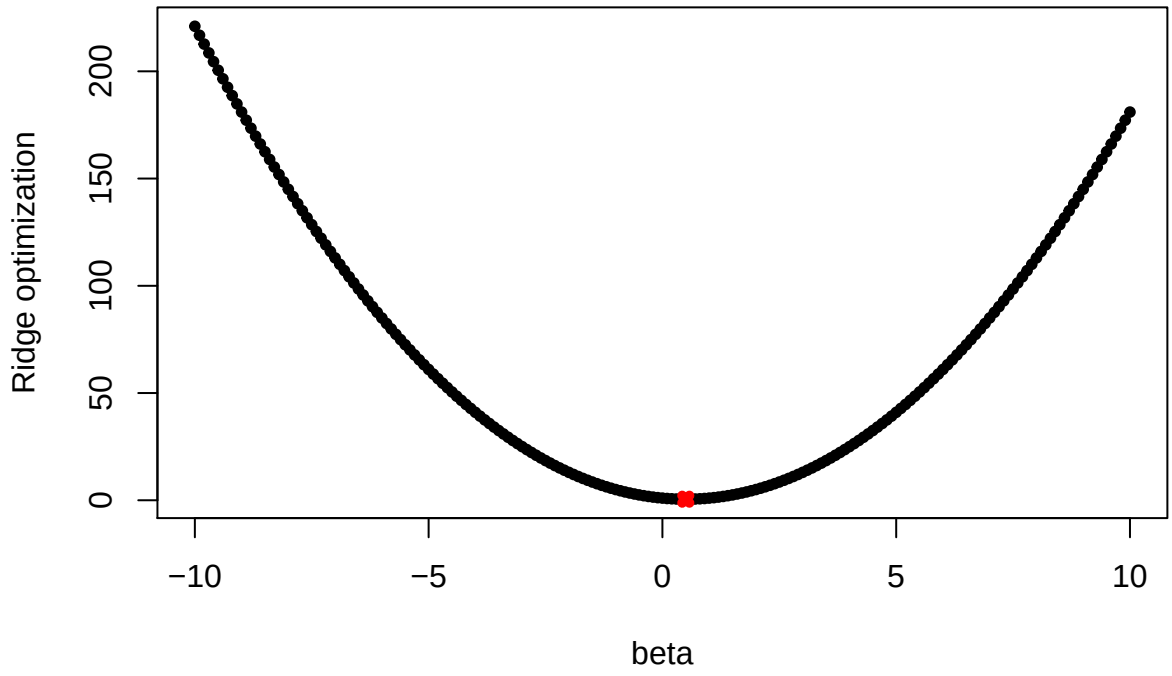
Xinwei Zhang xz2663

2/26/2018

P2 (a)

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

```
y = 1
lambda = 1
betas = seq(-10, 10, 0.1)
func = (y - betas)^2 + lambda * betas^2
plot(betas, func, pch=20, xlab="beta", ylab="Ridge optimization")
est.beta = y / (1+lambda)
est.func = (y - est.beta)^2 + lambda * est.beta^2
points(est.beta, est.func, col="red", pch=4, lwd=5, cex=est.beta)
```



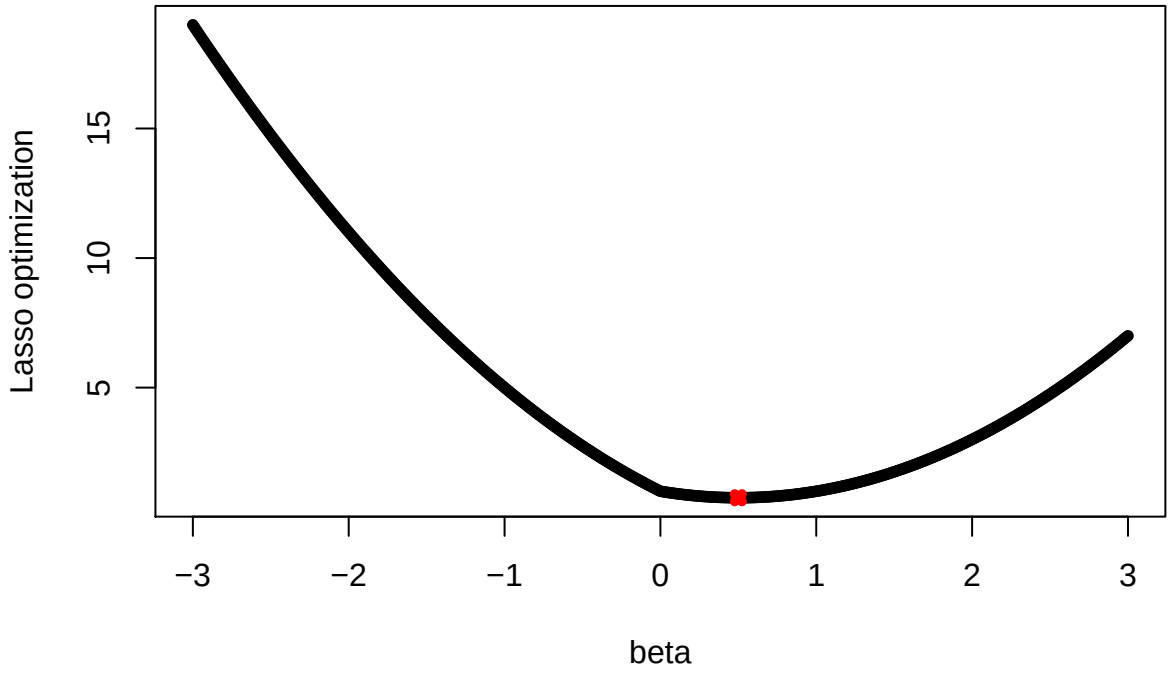
The red cross shows that function is minimized at $\beta = y/(1 + \lambda)$.

P2 (b)

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

```
y = 1
lambda = 1
betas = seq(-3, 3, 0.01)
func = (y - betas)^2 + lambda * abs(betas)
```

```
plot(betas, func, pch=20, xlab="beta", ylab="Lasso optimization")
est.beta = y - lambda / 2
est.func = (y - est.beta)^2 + lambda * abs(est.beta)
points(est.beta, est.func, col="red", pch=4, lwd=5, cex=est.beta)
```



The red cross shows that function is minimized at $\beta = y - \lambda/2$.

P4 (a)

Create X and ϵ

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

P4 (b)

We select $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = -3$ and $\beta_3 = 0.1$.

```
beta0 = 1
beta1 = 2
beta2 = -3
beta3 = 0.1
Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps
```

P4 (c)

```
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)

# Find the model size for best cp, BIC and adjr2
which.min(mod.summary$cp)
```

```
## [1] 3
```

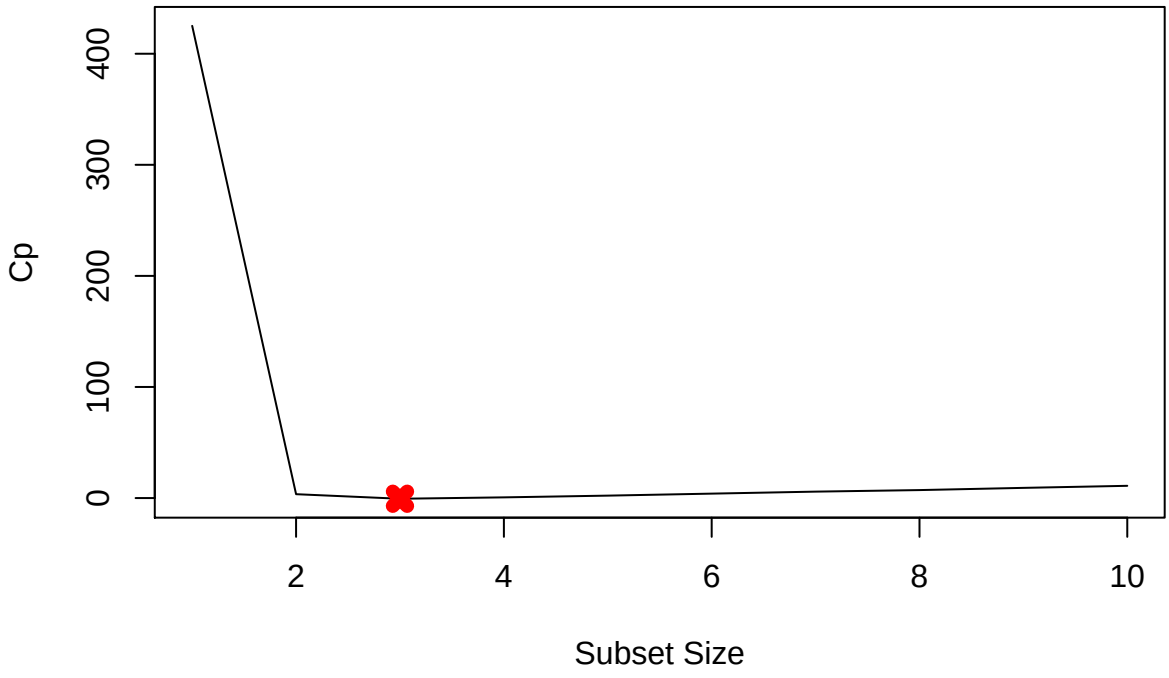
```
which.min(mod.summary$bic)
```

```
## [1] 3
```

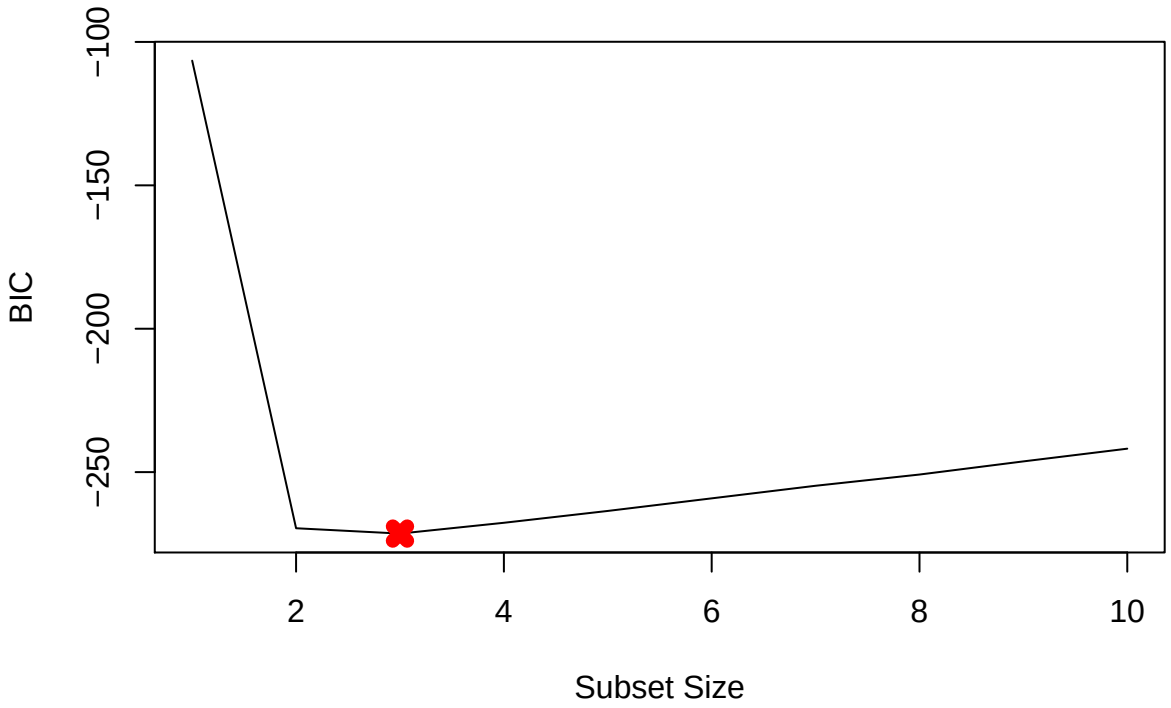
```
which.max(mod.summary$adjr2)
```

```
## [1] 3
```

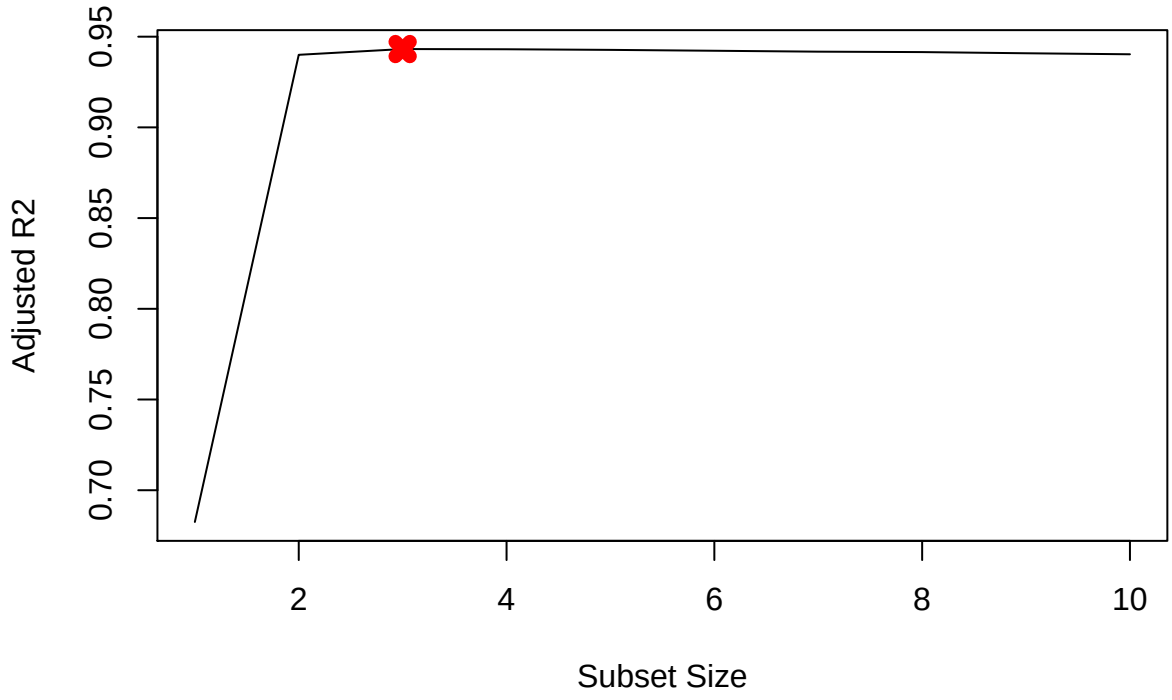
```
# Plot cp, BIC and adjr2
plot(mod.summary$cp, xlab="Subset Size", ylab="Cp", pch=20, type="l")
points(3, mod.summary$cp[3], pch=4, col="red", lwd=7)
```




```
plot(mod.summary$bic, xlab="Subset Size", ylab="BIC", pch=20, type="l")
points(3, mod.summary$bic[3], pch=4, col="red", lwd=7)
```




```
plot(mod.summary$adjr2, xlab="Subset Size", ylab="Adjusted R2", pch=20, type="l")
points(3, mod.summary$adjr2[3], pch=4, col="red", lwd=7)
```



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

```
coefficients(mod.full, 3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##           1.079475231           2.105467476          -3.159286180
## poly(x, 10, raw = T)9
##           0.000900817
```

All statistics pick X^9 over X^3. The remaining coefficients are close to β s.

P4 (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
```

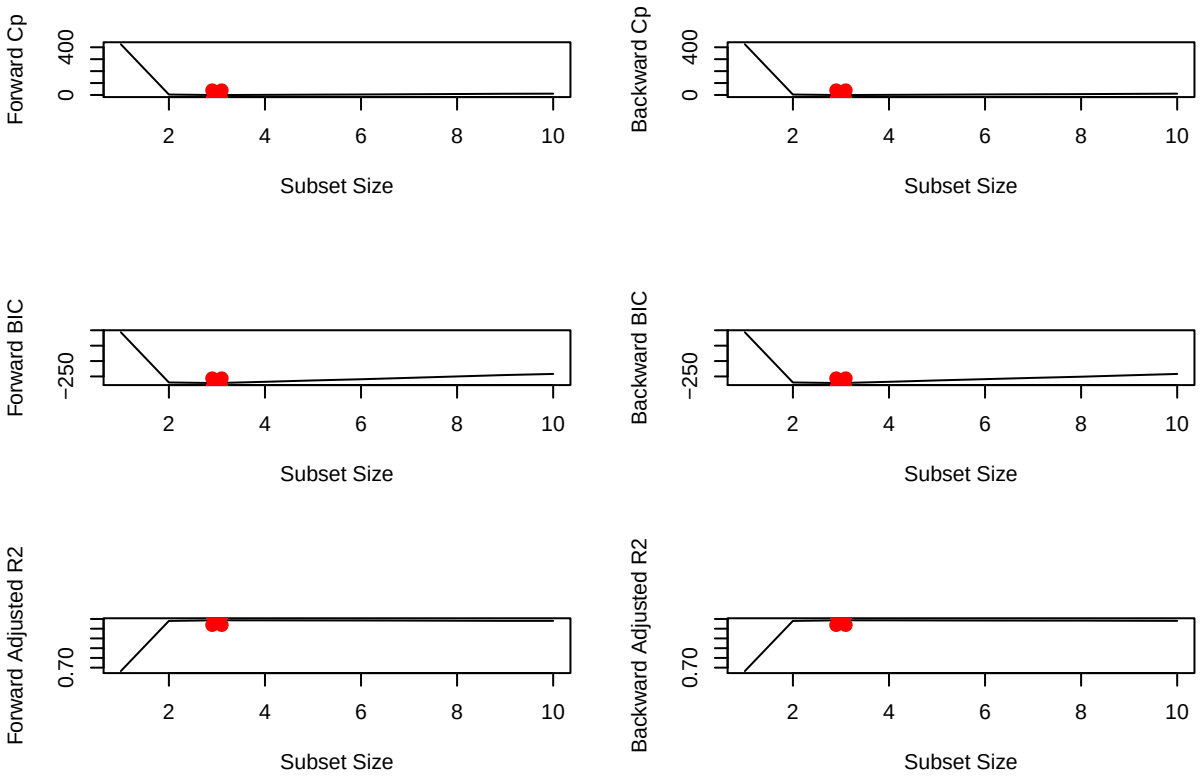
```
which.max(fwd.summary$adjr2)
```

```
## [1] 3
```

```
which.max(bwd.summary$adjr2)
```

```
## [1] 3
```

```
# Plot the statistics
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$adjr2, xlab="Subset Size", ylab="Backward Adjusted R2", pch=20, type="l")
points(3, bwd.summary$adjr2[3], pch=4, col="red", lwd=7)
```



We see that all statistics pick 3 variable models. Here are the coefficients:

```
coefficients(mod.fwd, id=3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##           1.079475231          2.105467476          -3.159286180
## poly(x, 10, raw = T)9
##           0.000900817
```

```
coefficients(mod.bwd, id=3)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##           1.079475231          2.105467476          -3.159286180
## poly(x, 10, raw = T)9
##           0.000900817
```

Forward stepwise picks X^9 over X^3. Backward stepwise picks X^9 over X^3. All other coefficients are close to β s.

P4 (e)

Training Lasso on the data

```
library(glmnet)
```

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

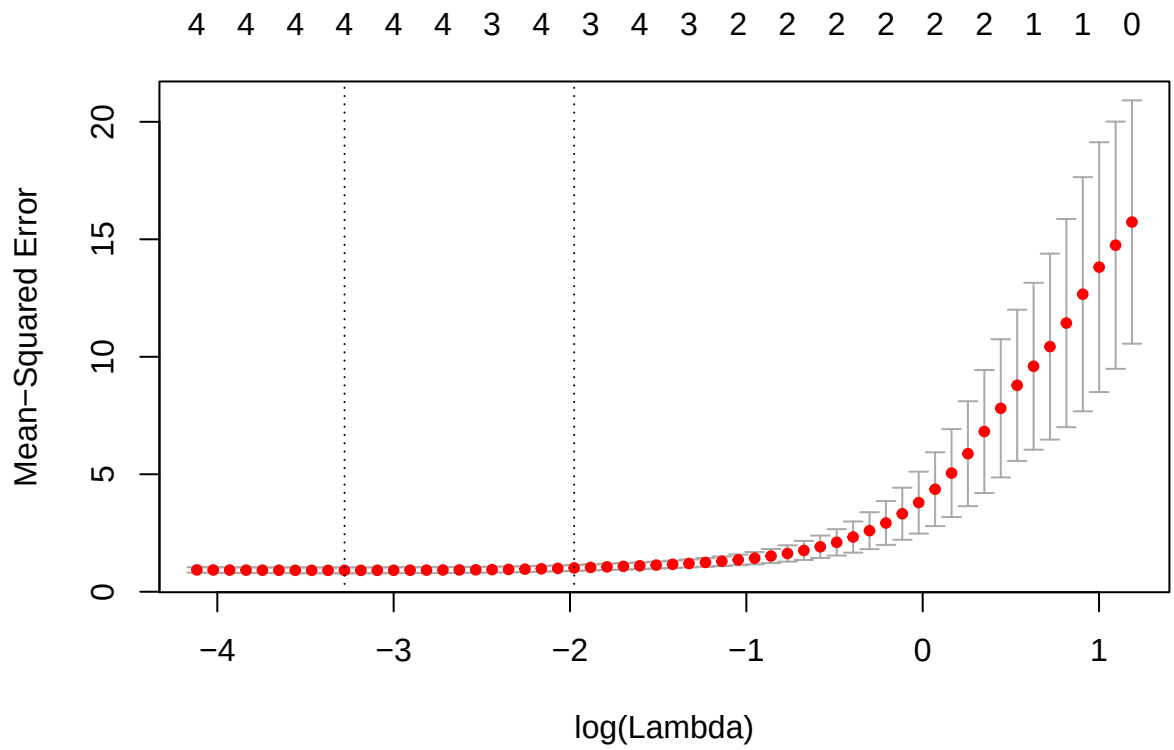
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
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] 0.03767797
```

```
plot(mod.lasso)
```




```
# 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)             1.0461279187
## poly(x, 10, raw = T)1   2.0730935473
## poly(x, 10, raw = T)2  -3.1096546561
## 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   0.0012341006
## poly(x, 10, raw = T)8   .
## poly(x, 10, raw = T)9   0.0005403851
## poly(x, 10, raw = T)10  .
```

Lasso picks X^7 over X^3. It also picks X^9 with negligible coefficient.

P4 (f)

Create new Y with different $\beta_7 = 7$.

```
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)

# Find the model size for 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
##             0.9589402             7.0007705
```

```
coefficients(mod.full, id=2)
```

```
##           (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
##             1.0704904            -0.1417084             7.0015552
```

```
coefficients(mod.full, id=4)
```

```
##           (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
##             1.0762524             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)            1.904188
## 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 (1.9 **vs** 1)**.**

P5 (a)

Load and split the College data

```
library(ISLR)
set.seed(11)
sum(is.na(College))
```

```
## [1] 0
```

```
train.size = dim(College)[1] / 2
train = sample(1:dim(College)[1], train.size)
test = -train
College.train = College[train, ]
College.test = College[test, ]
```

P5 (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

P5 (c)

```
library(glmnet)
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 1608859**.**

P5 (d)

```
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
```

Test RSS is 1635280**.**

The coefficients are

```
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
```

P6 (a)

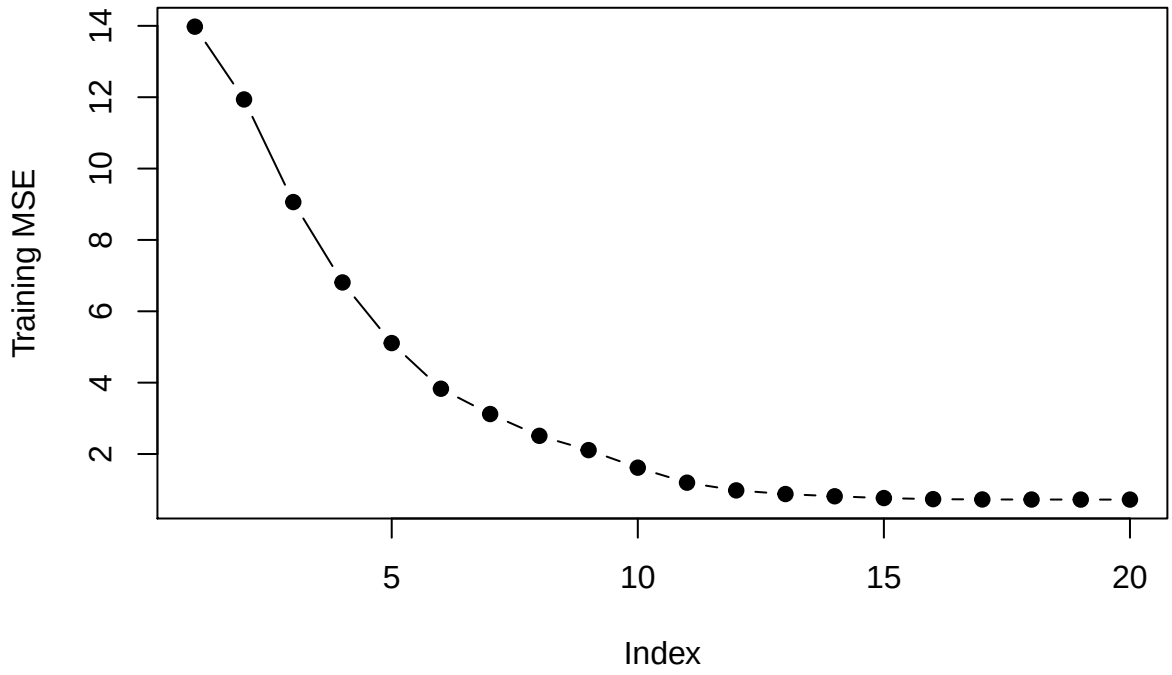
```
set.seed(1)
p = 20
n = 1000
x = matrix(rnorm(n*p), n, p)
B = rnorm(p)
B[1] = 0
B[5] = 0
B[13] = 0
B[3] = 0
B[17] = 0
eps = rnorm(p)
y = x %*% B + eps
```

P6 (b)

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

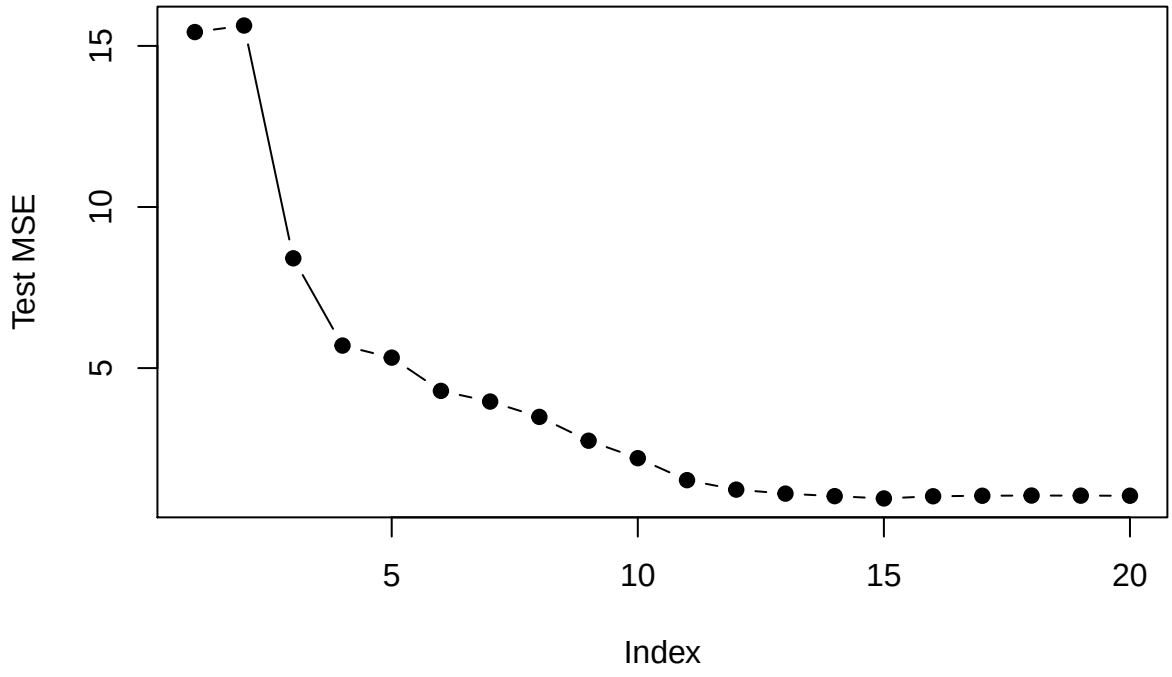
P6 (c)

```
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)
}
plot(val.errors, ylab="Training MSE", pch=19, type="b")
```



P6 (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)
}
plot(val.errors, ylab="Test MSE", pch=19, type="b")
```



P6 (e)

```
which.min(val.errors)
```

```
## [1] 15
```

15 parameter model has the smallest test MSE.

P6 (f)

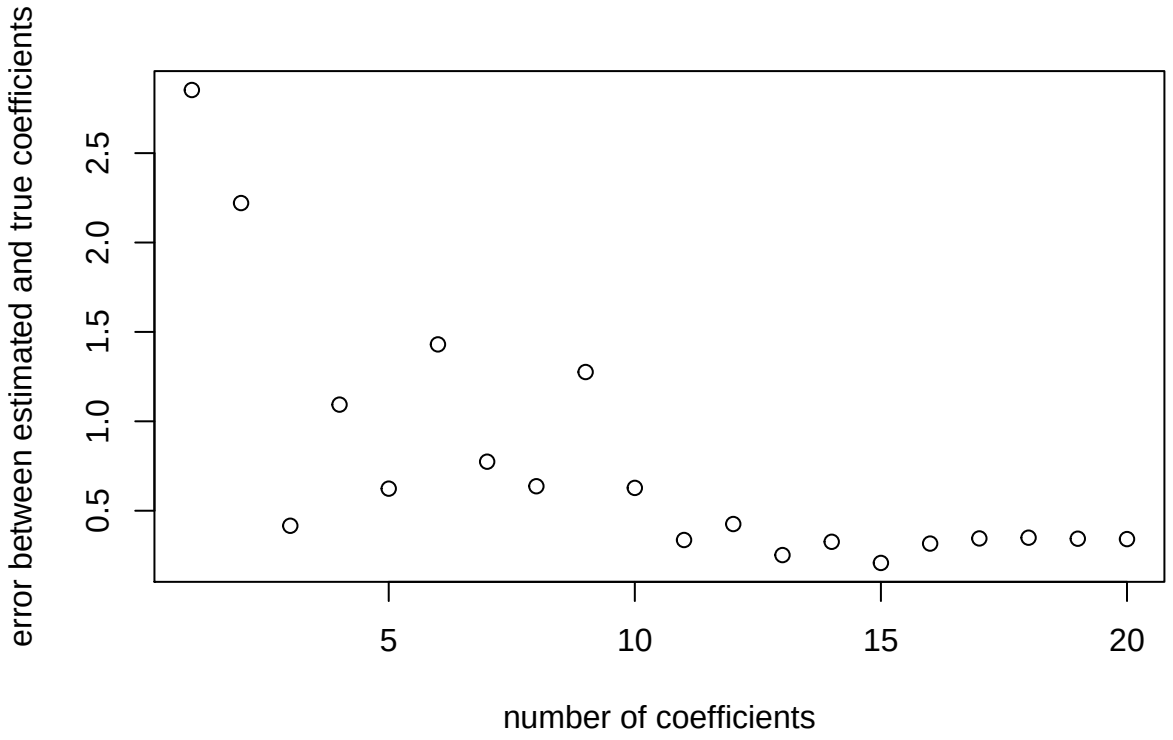
```
coef(regfit.full, id=15)
```

```
## (Intercept)         x.2         x.4         x.6         x.7         x.8
##  0.08822694  0.23806339 -1.83996077 -0.27840483 -1.48422745  0.76333155
##         x.9        x.10        x.11        x.12        x.14        x.15
##  2.01536530  0.68887253  0.91439492  0.49157906 -0.70893934 -0.73872394
##        x.16        x.18        x.19        x.20
## -0.29837212  1.76419486  0.80212121 -1.03276156
```

Caught all.

P6 (g)

```
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)
}
plot(x=a, y=b, xlab="number of coefficients",
     ylab="error between estimated and true coefficients")
```




```
which.min(b)
```

```
## [1] 15
```

Model with 15 coefficients minimizes the error between the estimated and true coefficients. Test error is minimized with 15 parameter model. It seems no relationship between good fit of true coefficients and low test MSE. A better fit of true coefficients as measured here do not mean the model will have a lower test MSE.