STK2100: Solutions Week 4

Lars H. B. Olsen

04.02.2021

ISLR 6.8.1

a)

Best subset selection has the smallest training RSS because the other two methods determine models with a path dependency on which predictors they pick first as they iterate to the k'th model.

b)

Best subset selection may have the smallest test RSS because it considers more models then the other methods. However, the other models might have better luck picking a model that fits the test data better. On the other hand, best subset could easily overfit if the data has large p predictors relative to n observations. Forward and backward selection might not converge on the same model but try the same number of models. It is hard to say which selection process would be better in general.

c)

1. True
2. True
3. False
4. False
5. False

ISLR 6.8.8

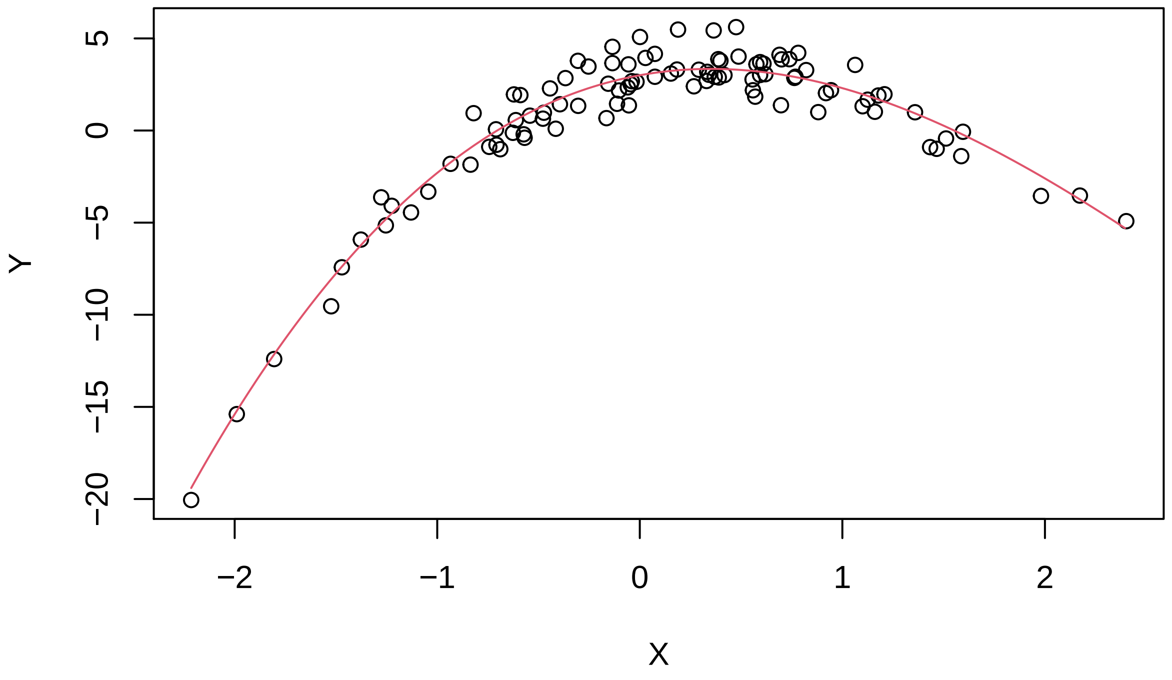
a)

```r
# Set seed fro reproducibility
set.seed(1)

# Set the length n
n = 100

# Create the predictor X of length n
X = rnorm(n)

# Create the noise vector epsilon of length n
eps = rnorm(100)
```

b)

We are free to chose the constants β_0, β_1, β_2, and β_3. We let them be 3, 2, -3, and 0.3, respectively.

```r
# Set the true coefficients
beta0 = 3
beta1 = 2
beta2 = -3
beta3 = 0.3

# Generate the response vector
Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps

# Plot the observed data
plot(X, Y)
xx_values = seq(min(X), max(X), by = 0.01)
yy_values = beta0 + beta1 * xx_values + beta2 * xx_values^2 + beta3 * xx_values^3
lines(xx_values, yy_values, col = 2)
```



c)

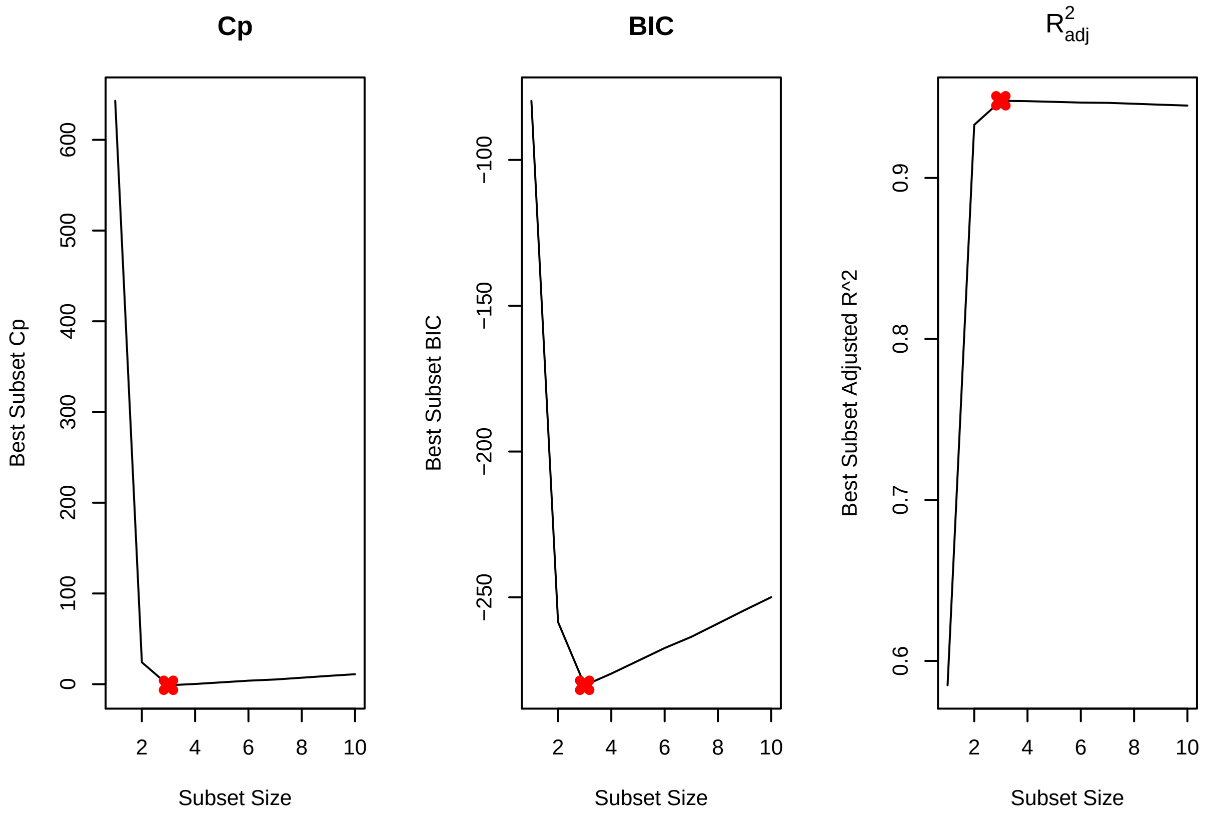
Here we are using `regsubsets()` to perform best-subset selection to chose the best model containing the predictors X, X^2,..., X^{10}.

```r
# Use regsubsets to select best model having polynomial of X of degree 10
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)

par(mfrow=c(1,3))

# Find the model size for best Cp, BIC and adjr2.
min_cp_model = which.min(mod.summary$cp)
min_bic_model = which.min(mod.summary$bic)
min_adjr2_model = which.max(mod.summary$adjr2)
```

```r
# The all say that the three parameter model is the best
show(cbind(min_cp_model, min_bic_model, min_adjr2_model))
```

```
##      min_cp_model min_bic_model min_adjr2_model
## [1,]            3             3               3
```

```r
# Plot Cp, BIC and adjr2
plot(mod.summary$cp, xlab="Subset Size", ylab="Best Subset Cp",
     type="l", main = "Cp")
points(min_cp_model, mod.summary$cp[min_cp_model], col="red", pch=4, lwd=5)

plot(mod.summary$bic, xlab="Subset Size", ylab="Best Subset BIC",
     type="l", main = "BIC")
points(min_bic_model, mod.summary$bic[min_bic_model], col="red", pch=4, lwd=5)

plot(mod.summary$adjr2, xlab="Subset Size", ylab="Best Subset Adjusted R^2",
     type="l", main = bquote(R["adj"]^2))
points(min_adjr2_model, mod.summary$adjr2[min_adjr2_model], col="red", pch=4, lwd=5)
```



```r
# All three model choose a three parameter model,
# but neither model includes the thrid degree polinomial.
# They all go for X, X^2, and x^7
coef(mod.full, id = min_cp_model)
```

```
##           (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
```

```r
coef(mod.full, id = min_bic_model)
```

```
##           (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
```

```r
coef(mod.full, id = min_adjr2_model)
```

```
##           (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_0, β_1, and β_2.

d)

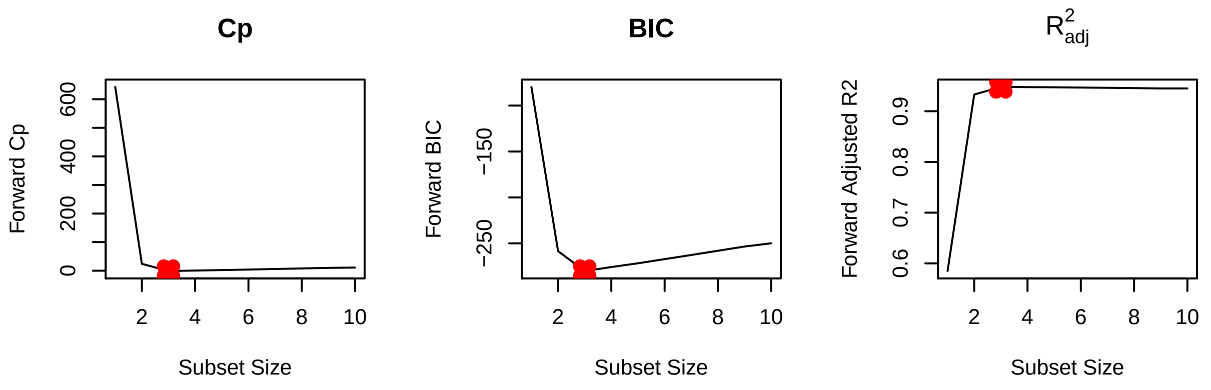
We fit forward and backward stepwise models to the data.

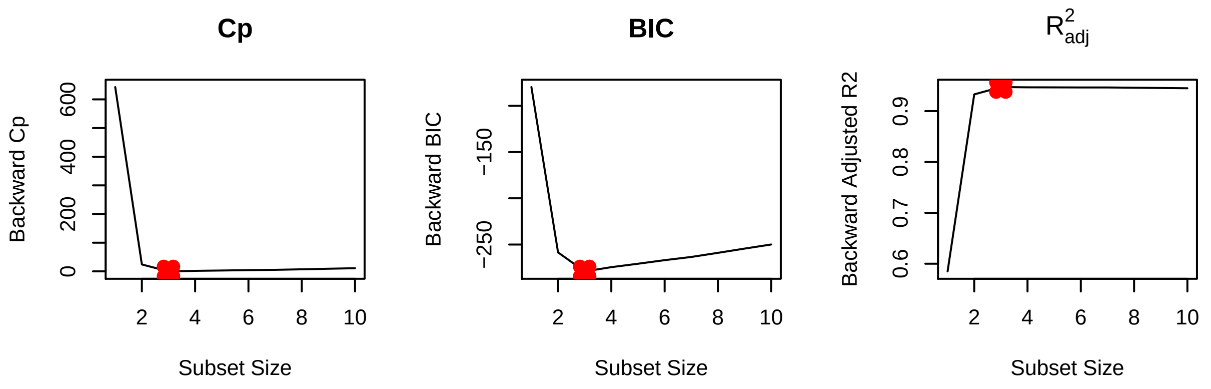
```r
# We fit forward stepwise models to the data.
mod.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,
                     method = "forward")
fwd.summary = summary(mod.fwd)

# We fit backward stepwise models to the data.
mod.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,
                     method = "backward")
bwd.summary = summary(mod.bwd)

par(mfrow=c(3,2))

# Find the best Cp, BIC, adjr2 statistics
best_values = matrix(c(which.min(fwd.summary$cp), which.min(bwd.summary$cp),
                       which.min(fwd.summary$bic), which.min(bwd.summary$bic),
                       which.max(fwd.summary$adjr2), which.max(bwd.summary$adjr2)),
                     ncol = 2, byrow = TRUE)
rownames(best_values) = c("Cp", "BIC", "adjR2")
colnames(best_values) = c("Forward", "Backward")
show(t(best_values))
```

```
##          Cp BIC adjR2
## Forward   3   3     3
## Backward  3   3     3
```

```r
## Plot the statistics
par(mfrow = c(2, 3))

# Forward
plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward Cp", pch = 20,
     type = "l", main = "Cp")
points(3, fwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

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

```r
plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R2",
     pch = 20, type = "l", main = bquote(R["adj"]^2))
points(3, fwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)

# backward
plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward Cp", pch = 20,
     type = "l", main = "Cp")
points(3, bwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

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

plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R2",
     pch = 20, type = "l", main = bquote(R["adj"]^2))
points(3, bwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
```





```r
# We see that all statistics pick 3 variable. Here are the coefficients
coef(mod.fwd, which.min(fwd.summary$cp))
```

```
##           (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
```

```r
coef(mod.bwd, which.min(bwd.summary$cp))
```

```
##           (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
```

```
coef(mod.fwd, which.min(fwd.summary$bic))
```

```
##          (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
```

```
coef(mod.bwd, which.min(bwd.summary$bic))
```

```
##          (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
```

```
coef(mod.fwd, which.max(fwd.summary$adjr2))
```

```
##          (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
```

```
coef(mod.bwd, which.max(bwd.summary$adjr2))
```

```
##          (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
```

Here we see that forward stepwise picks X^7 over X^3. Backward stepwise, on the other hand, picks X^9 insted of X^3.

Body fat data

```
# Include 'MASS' library that has the 'stepAIC' function
library(MASS)

# Download the data
path = "https://www.uio.no/studier/emner/matnat/math/STK2100/v21/data/edu_bodyfat.csv"
data = read.csv(path, header = TRUE, row.names = 1)

# Look at the data
head(data)
```

```
##   age weight height neck chest    ab   hip thigh knee ankle biceps forearm
## 1  23 154.25  67.75 36.2  93.1  85.2  94.5  59.0 37.3  21.9   32.0    27.4
## 2  22 173.25  72.25 38.5  93.6  83.0  98.7  58.7 37.3  23.4   30.5    28.9
## 3  22 154.00  66.25 34.0  95.8  87.9  99.2  59.6 38.9  24.0   28.8    25.2
## 4  26 184.75  72.25 37.4 101.8  86.4 101.2  60.1 37.3  22.8   32.4    29.4
## 5  24 184.25  71.25 34.4  97.3 100.0 101.9  63.2 42.2  24.0   32.2    27.7
## 6  24 210.25  74.75 39.0 104.5  94.4 107.8  66.0 42.0  25.6   35.7    30.6
##   wrist pcfat
## 1  17.1  12.3
## 2  18.2   6.1
## 3  16.6  25.3
## 4  18.2  10.4
## 5  17.7  28.7
## 6  18.8  20.9
```

```
# Define the intercept model and the full model
model_wide = lm(pcfat ~ ., data = data)
model_narow = lm(pcfat ~ 1, data = data)

# See documentation
?stepAIC

# Then use stepAIC to conduct forward elimination
model_backward_AIC = stepAIC(model_wide, direction="backward",
                             k = 2, data=data, trace = TRUE)
```

```
## Start:  AIC=749.85
## pcfat ~ age + weight + height + neck + chest + ab + hip + thigh +
##     knee + ankle + biceps + forearm + wrist
##
##           Df Sum of Sq    RSS    AIC
## - knee     1      0.06 4420.1 747.85
## - chest    1      0.51 4420.6 747.88
## - height   1      1.12 4421.2 747.91
## - ankle    1     11.86 4431.9 748.52
## - biceps   1     20.74 4440.8 749.03
## - hip      1     31.51 4451.6 749.64
## <none>                 4420.1 749.85
## - weight   1     45.10 4465.2 750.41
## - thigh    1     53.61 4473.7 750.89
## - age      1     74.72 4494.8 752.07
## - neck     1     75.66 4495.7 752.13
## - forearm  1     97.11 4517.2 753.33
## - wrist    1    178.85 4598.9 757.84
## - ab       1   2083.46 6503.5 845.17
##
## Step:  AIC=747.85
## pcfat ~ age + weight + height + neck + chest + ab + hip + thigh +
##     ankle + biceps + forearm + wrist
##
##           Df Sum of Sq    RSS    AIC
## - chest    1      0.52 4420.6 745.88
## - height   1      1.06 4421.2 745.91
## - ankle    1     12.59 4432.7 746.57
## - biceps   1     20.68 4440.8 747.03
## - hip      1     31.47 4451.6 747.64
## <none>                 4420.1 747.85
## - weight   1     45.26 4465.4 748.42
## - thigh    1     60.46 4480.6 749.28
## - neck     1     77.09 4497.2 750.21
## - age      1     80.99 4501.1 750.43
## - forearm  1     98.18 4518.3 751.39
## - wrist    1    179.35 4599.5 755.88
## - ab       1   2083.40 6503.5 843.17
##
```

```
## Step:  AIC=745.88
## pcfat ~ age + weight + height + neck + ab + hip + thigh + ankle +
##     biceps + forearm + wrist
##
##           Df Sum of Sq    RSS    AIC
## - height   1      0.68 4421.3 743.92
## - ankle    1     12.90 4433.5 744.62
## - biceps   1     20.44 4441.1 745.04
## - hip      1     31.11 4451.8 745.65
## <none>                 4420.6 745.88
## - weight   1     64.84 4485.5 747.55
## - thigh    1     65.82 4486.5 747.61
## - neck     1     76.90 4497.5 748.23
## - age      1     80.68 4501.3 748.44
## - forearm  1     97.89 4518.5 749.40
## - wrist    1    178.96 4599.6 753.88
## - ab       1   2350.68 6771.3 851.34
##
## Step:  AIC=743.92
## pcfat ~ age + weight + neck + ab + hip + thigh + ankle + biceps +
##     forearm + wrist
##
##           Df Sum of Sq    RSS    AIC
## - ankle    1      13.3 4434.6 742.68
## - biceps   1      22.4 4443.7 743.19
## - hip      1      30.4 4451.8 743.65
## <none>                 4421.3 743.92
## - thigh    1      68.8 4490.1 745.81
## - neck     1      77.1 4498.4 746.28
## - age      1      81.3 4502.6 746.51
## - forearm  1      98.1 4519.4 747.45
## - weight   1     119.6 4540.9 748.65
## - wrist    1     181.3 4602.6 752.05
## - ab       1    3178.5 7599.9 878.43
##
## Step:  AIC=742.68
## pcfat ~ age + weight + neck + ab + hip + thigh + biceps + forearm +
##     wrist
##
##           Df Sum of Sq    RSS    AIC
## - biceps   1      20.7 4455.3 741.85
## - hip      1      31.7 4466.4 742.47
## <none>                 4434.6 742.68
## - thigh    1      72.3 4506.9 744.75
## - age      1      77.6 4512.2 745.05
## - neck     1      87.3 4521.9 745.59
## - forearm  1      97.4 4532.0 746.15
## - weight   1     107.2 4541.8 746.69
## - wrist    1     168.0 4602.6 750.05
## - ab       1    3182.0 7616.7 876.98
##
## Step:  AIC=741.85
## pcfat ~ age + weight + neck + ab + hip + thigh + forearm + wrist
##
```

```
##           Df Sum of Sq    RSS    AIC
## <none>                 4455.3 741.85
## - hip      1      36.5 4491.8 741.91
## - neck     1      79.1 4534.4 744.29
## - age      1      83.8 4539.1 744.55
## - weight   1      93.0 4548.3 745.05
## - thigh    1     100.7 4556.0 745.48
## - forearm  1     140.5 4595.8 747.67
## - wrist    1     166.8 4622.2 749.12
## - ab       1    3163.0 7618.3 875.04
```

```
# Then use stepAIC to conduct backard elimination
model_forward_AIC = stepAIC(model_narow, direction="forward", k = 2,
                            scope = list(lower = model_narow, upper = model_wide))
```

```
## Start:  AIC=1071.75
## pcfat ~ 1
##
##           Df Sum of Sq     RSS     AIC
## + ab       1   11631.5  5947.5  800.65
## + chest    1    8678.3  8900.7  902.24
## + hip      1    6871.2 10707.8  948.82
## + weight   1    6593.0 10986.0  955.29
## + thigh    1    5505.0 12073.9  979.08
## + knee     1    4548.4 13030.6  998.30
## + biceps   1    4277.3 13301.7 1003.49
## + neck     1    4230.9 13348.1 1004.36
## + forearm  1    2295.8 15283.2 1038.48
## + wrist    1    2111.5 15467.5 1041.50
## + age      1    1493.3 16085.7 1051.38
## + ankle    1    1243.5 16335.5 1055.26
## <none>                 17579.0 1071.75
## + height   1      11.2 17567.7 1073.59
##
## Step:  AIC=800.65
## pcfat ~ ab
##
##           Df Sum of Sq    RSS    AIC
## + weight   1   1004.22 4943.2 756.04
## + wrist    1    709.18 5238.3 770.65
## + neck     1    614.52 5332.9 775.16
## + height   1    588.53 5358.9 776.39
## + hip      1    548.24 5399.2 778.27
## + knee     1    318.70 5628.8 788.77
## + ankle    1    233.32 5714.1 792.56
## + age      1    200.93 5746.5 793.98
## + chest    1    195.45 5752.0 794.22
## + thigh    1    174.58 5772.9 795.14
## + biceps   1    135.30 5812.2 796.85
## + forearm  1     54.30 5893.2 800.33
## <none>                 5947.5 800.65
##
## Step:  AIC=756.04
## pcfat ~ ab + weight
##
```

```
##           Df Sum of Sq    RSS    AIC
## + wrist    1   157.191 4786.1 749.90
## + neck     1    86.929 4856.3 753.57
## + thigh    1    81.356 4861.9 753.86
## + forearm  1    66.852 4876.4 754.61
## + biceps   1    63.809 4879.4 754.77
## <none>                 4943.2 756.04
## + knee     1     9.719 4933.5 757.54
## + height   1     7.284 4936.0 757.67
## + age      1     1.938 4941.3 757.94
## + ankle    1     1.514 4941.7 757.96
## + chest    1     0.010 4943.2 758.04
## + hip      1     0.005 4943.2 758.04
##
## Step:  AIC=749.9
## pcfat ~ ab + weight + wrist
##
##           Df Sum of Sq    RSS    AIC
## + forearm  1   127.819 4658.2 745.07
## + biceps   1    88.731 4697.3 747.18
## + thigh    1    40.463 4745.6 749.76
## <none>                 4786.1 749.90
## + neck     1    25.183 4760.9 750.57
## + age      1    21.153 4764.9 750.78
## + knee     1    20.545 4765.5 750.81
## + ankle    1    14.970 4771.1 751.11
## + hip      1     9.229 4776.8 751.41
## + height   1     5.916 4780.1 751.58
## + chest    1     1.257 4784.8 751.83
##
## Step:  AIC=745.07
## pcfat ~ ab + weight + wrist + forearm
##
##          Df Sum of Sq    RSS    AIC
## + neck    1    51.066 4607.2 744.30
## + age     1    38.362 4619.9 744.99
## <none>                4658.2 745.07
## + biceps  1    33.883 4624.4 745.23
## + thigh   1    27.215 4631.0 745.60
## + knee    1    19.829 4638.4 746.00
## + ankle   1    18.158 4640.1 746.09
## + hip     1     3.532 4654.7 746.88
## + height  1     1.703 4656.5 746.98
## + chest   1     0.488 4657.7 747.05
##
## Step:  AIC=744.3
## pcfat ~ ab + weight + wrist + forearm + neck
##
##          Df Sum of Sq    RSS    AIC
## + age     1    47.934 4559.2 743.66
## + biceps  1    45.927 4561.2 743.77
## <none>                4607.2 744.30
## + thigh   1    25.101 4582.1 744.92
## + hip     1    10.995 4596.2 745.69
## + ankle   1    10.665 4596.5 745.71
## + knee    1    10.396 4596.8 745.73
## + height  1     6.700 4600.5 745.93
## + chest   1     0.010 4607.2 746.30
##
## Step:  AIC=743.66
## pcfat ~ ab + weight + wrist + forearm + neck + age
##
##          Df Sum of Sq    RSS    AIC
## + thigh   1    67.387 4491.8 741.91
## + biceps  1    48.138 4511.1 742.99
## <none>                4559.2 743.66
## + ankle   1    14.776 4544.5 744.84
## + height  1     9.357 4549.9 745.14
## + knee    1     6.552 4552.7 745.30
## + hip     1     3.230 4556.0 745.48
## + chest   1     0.843 4558.4 745.61
##
## Step:  AIC=741.91
## pcfat ~ ab + weight + wrist + forearm + neck + age + thigh
##
##          Df Sum of Sq    RSS    AIC
## + hip     1    36.524 4455.3 741.85
## <none>                4491.8 741.91
## + biceps  1    25.488 4466.4 742.47
## + ankle   1    12.768 4479.1 743.19
## + height  1     0.941 4490.9 743.86
## + chest   1     0.755 4491.1 743.87
## + knee    1     0.003 4491.8 743.91
##
## Step:  AIC=741.85
## pcfat ~ ab + weight + wrist + forearm + neck + age + thigh +
##     hip
##
##          Df Sum of Sq    RSS    AIC
## <none>                4455.3 741.85
## + biceps  1   20.7116 4434.6 742.68
## + ankle   1   11.6200 4443.7 743.19
## + height  1    3.1643 4452.2 743.67
## + knee    1    0.0365 4455.3 743.85
## + chest   1    0.0001 4455.3 743.85
```

```
# Look at the summary of the models
sum_model_backward_AIC = summary(model_backward_AIC)
sum_model_forward_AIC = summary(model_forward_AIC)

# We see that both models contain the same explantory variables.
show(rbind(sort(attr(sum_model_backward_AIC$terms, "term.labels")),
           sort(attr(sum_model_forward_AIC$terms, "term.labels"))))
```

```
##      [,1] [,2]  [,3]      [,4]  [,5]   [,6]    [,7]     [,8]
## [1,] "ab" "age" "forearm" "hip" "neck" "thigh" "weight" "wrist"
## [2,] "ab" "age" "forearm" "hip" "neck" "thigh" "weight" "wrist"
```

```
# Look at the best model according to AIC
# found by both forward and backward elimination.
sum_model_backward_AIC
```

```
##
## Call:
## lm(formula = pcfat ~ age + weight + neck + ab + hip + thigh +
##     forearm + wrist, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9757  -2.9937  -0.1644   2.9767  10.2244
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -22.65638   11.71386  -1.934  0.05426 .
## age           0.06578    0.03078   2.137  0.03356 *
## weight       -0.08985    0.03991  -2.252  0.02524 *
## neck         -0.46656    0.22462  -2.077  0.03884 *
## ab            0.94482    0.07193  13.134  < 2e-16 ***
## hip          -0.19543    0.13847  -1.411  0.15940
## thigh         0.30239    0.12904   2.343  0.01992 *
## forearm       0.51572    0.18631   2.768  0.00607 **
## wrist        -1.53665    0.50939  -3.017  0.00283 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.282 on 243 degrees of freedom
## Multiple R-squared:  0.7466, Adjusted R-squared:  0.7382
## F-statistic: 89.47 on 8 and 243 DF,  p-value: < 2.2e-16
```

Hitters Ridge Regression

Also see Section 6.5 in ISLR.

```
# Get the datasets from the book
library(ISLR)

# Look at the names of the explanatory variables
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```
# Look at the dimension of the Hitters data set
dim(Hitters) # 322 x 20
```

```
## [1] 322  20
```

```
# See number of observations with missing salaries
sum(is.na(Hitters$Salary)) # 50
```

```
## [1] 59
```

```r
# Remove all instances with anything missing
Hitters = na.omit(Hitters) # In total 59

# Look at the new dimensions
dim(Hitters) # 263  20
```

```
## [1] 263  20
```

```r
# See if data set contains any missing entries
sum(is.na(Hitters)) # 0
```

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

```r
# Copy the data set
Hitters2 = Hitters

# Normalize 'PutOuts' and 'Hits',
# such that the have expectation 0 and variance 1
Hitters2$PutOuts = (Hitters2$PutOuts - mean(Hitters2$PutOuts)) / sd(Hitters2$PutOuts)
Hitters2$Hits = (Hitters2$Hits-mean(Hitters2$Hits)) / sd(Hitters2$Hits)
mean(Hitters2$Hits) # approx 0.
```

```
## [1] 7.132401e-17
```

```r
var(Hitters2$Hits)  # = 1
```

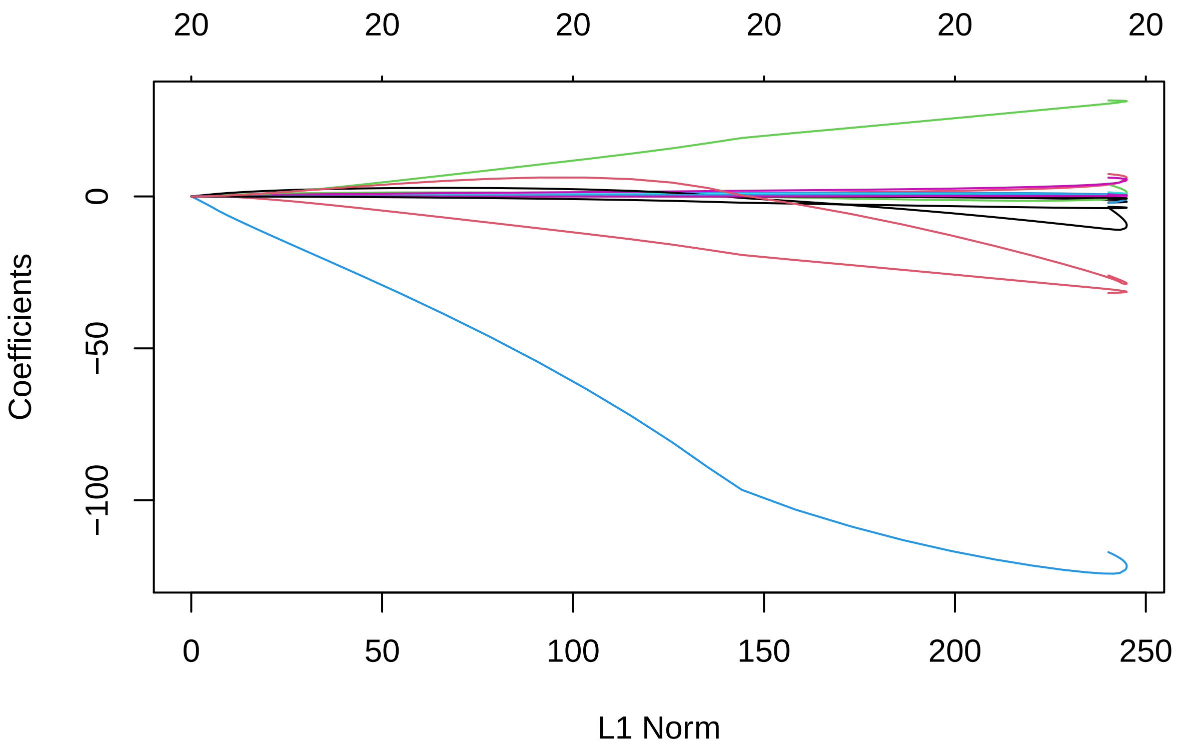
```
## [1] 1
```

```r
# Fit a linear model
fit.lm = lm(Salary ~ PutOuts + Hits, data = Hitters2)

# Creates the design matrix
?model.matrix
x = model.matrix(Salary ~ . - 1, Hitters)

# Define the reponse values
y = Hitters$Salary

# For understanding the following commands, you can look at the help pages or
# Look at section 6.5 in the ISLR book (James et al)

# Load the glmnet library
# Needed to fit lasso, ridge, or elastic models
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1
```

```r
?glmnet

# Create a sequence of lambda values used in elastic net
grid = 10^seq(10,-2,length=100)

# Fit a ridge model
ridge.mod=glmnet(x, y, alpha=0, lambda=grid)

# Plot the coefficient profile plot
```

```r
plot(ridge.mod)
```



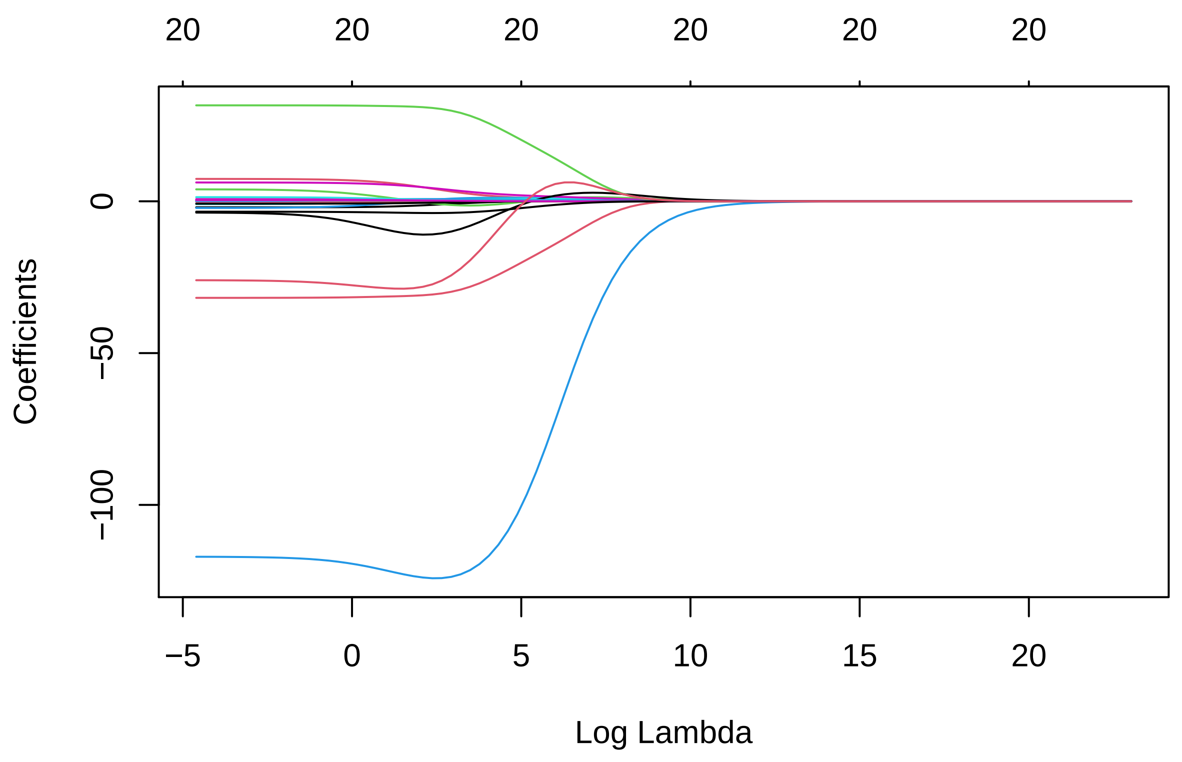
```r
# Look at class of object ridge.mod
?class
class(ridge.mod)
```

```
## [1] "elnet"  "glmnet"
```

```r
# Look at help functions descibing what we plotted above
help(plot.elnet)
```

```
## No documentation for 'plot.elnet' in specified packages and libraries:
## you could try '??plot.elnet'
```

```r
help(plot.glmnet)

# Create a new coefficient profile plot with different values on the x axis
plot(ridge.mod, xvar="lambda")
```



```r
# The range of the x-axis in figure above
range(log(grid))
```

```
## [1] -4.60517 23.02585
```

```r
# Get the dimension of the coeficients computed by glmnet for the different
# lambda values.
# We have 100 lambda values, and 21 explanatory variables (20 + intercept).
dim(coef(ridge.mod))
```

```
## [1]  21 100
```

```r
# Look at the 50th lambda value. Same as grid[50].
ridge.mod$lambda[50] # = 11497.57
```

```
## [1] 11497.57
```

```r
# Look at the model coeffcent corresponding to using the lambda above
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 407.399396904   0.036958511   0.138184676   0.524663826   0.230712132
##           RBI         Walks         Years        CAtBat         CHits
##   0.239850635   0.289621892   1.107710345   0.003131822   0.011653657
##        CHmRun         CRuns          CRBI        CWalks       LeagueA
##   0.087547281   0.023380056   0.024138479   0.025015505  -0.082038139
##       LeagueN     DivisionW       PutOuts       Assists        Errors
##   0.082040144  -6.215425597   0.016482325   0.002612463  -0.020523535
##    NewLeagueN
##   0.298876792
```

```r
# Look at square root of the sum of squared model coefficients
sqrt(sum(coef(ridge.mod)[-1,50]^2)) # = 6.360972
```

```
## [1] 6.360972
```

```
# Then we do the same for the 60th lambda value
ridge.mod$lambda[60] # = 705.4802
```

```
## [1] 705.4802
```

```
# We now get larger coefficients as we have a smaller penalty
coef(ridge.mod)[,60]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs           RBI
##   61.72599501    0.11266871    0.65790290    1.19513175    0.94289598    0.85041323
##         Walks         Years        CAtBat         CHits        CHmRun         CRuns
##    1.31891024    2.60105804    0.01082533    0.04671252    0.33835636    0.09359447
##          CRBI        CWalks       LeagueA       LeagueN     DivisionW       PutOuts
##    0.09777821    0.07182962 -10.49032970   10.48716517  -54.62727686    0.11823107
##       Assists        Errors    NewLeagueN
##    0.01577698   -0.72133826    6.22986245
```

```
# See that this quantity has increased do to larger model coefficients.
sqrt(sum(coef(ridge.mod)[-1,60]^2)) # = 57.05825
```

```
## [1] 57.05825
```

```
# Computes the coefficients at the requested values for s
?predict.glmnet
predict(ridge.mod, s=50, type="coefficients")[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  7.364405e+01 -3.566913e-01  1.970091e+00 -1.271214e+00  1.151250e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.044928e-01  2.713730e+00 -6.197479e+00  5.435118e-03  1.065094e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueA
##  6.247569e-01  2.216810e-01  2.184314e-01 -1.503916e-01 -2.635203e+01
##       LeagueN     DivisionW       PutOuts       Assists        Errors
##  2.636396e+01 -1.181501e+02  2.499668e-01  1.215336e-01 -3.311239e+00
```

```
# Set seed for reproducibility
set.seed(1)

# Get indices for training and test data
train = sample(1:nrow(x), nrow(x)/2)
test = -train

# Create the test response
y.test = y[test]

# Fit a ridge model to the training data
ridge.mod = glmnet(x[train,], y[train], alpha=0, lambda=grid, thresh=1e-12)

# Predict the repose using the ridge model with lambda = 4
ridge.pred=predict(ridge.mod, s=4, newx=x[test,])

# Compute the mean RSS and mean ESS
mean((ridge.pred-y.test)^2) # = 142147.4
```

```
## [1] 142147.4
```

```
mean((mean(y[train])-y.test)^2) # = 224669.9
```

```
## [1] 224669.9
```

```
# Predict test set, now with a larger penalty. Get a larger mean RSS
ridge.pred=predict(ridge.mod, s=1e10, newx=x[test,])
mean((ridge.pred-y.test)^2) # = 224669.8
```

```
## [1] 224669.8
```

```
# Predict response. See ?coef.glmnet for meaninig of parameters
ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train])
mean((ridge.pred-y.test)^2) # = 168588.6
```

```
## [1] 168588.6
```

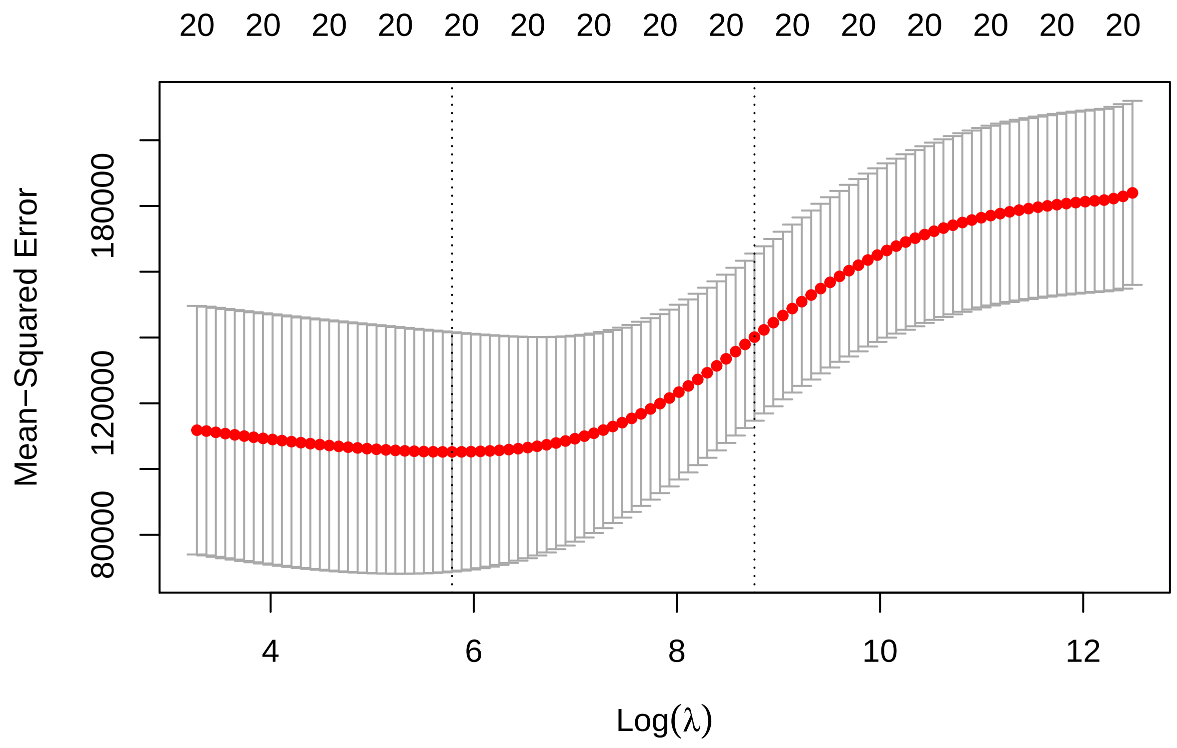
```
# Fit a linear model to the training data
lm(y~x, subset=train)
```

```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
## (Intercept)       xAtBat        xHits       xHmRun        xRuns         xRBI
##    393.1631      -0.3521      -1.6377       5.8145       1.5424       1.1243
##      xWalks       xYears      xCAtBat       xCHits      xCHmRun       xCRuns
##      3.7287     -16.3773      -0.6412       3.1632       3.4008      -0.9739
##       xCRBI      xCWalks     xLeagueA     xLeagueN    xDivisionW     xPutOuts
##     -0.6005       0.3379    -119.1486           NA     -144.0831       0.1976
##    xAssists      xErrors  xNewLeagueN
##      0.6804      -4.7128     -71.0951
```

```
predict(ridge.mod,s=0,exact=T,x=x[train,],y=y[train],type="coefficients")[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##  335.2748639   -0.3521900   -1.6371384    5.8146691    1.5423362    1.1241838
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##    3.7288406  -16.3795193   -0.6411235    3.1629444    3.4005281   -0.9739405
##         CRBI       CWalks      LeagueA      LeagueN    DivisionW      PutOuts
##   -0.6003976    0.3378422  -61.2547650   57.8886991 -144.0853059    0.1976300
##      Assists       Errors
##    0.6804200   -4.7127879
```

```
# Set seed for reproducibility
set.seed(1)

# Do cross-validation
cv.out=cv.glmnet(x[train,],y[train],alpha=0)

# Look at MSE for different baulues of log(lambda).
plot(cv.out)
```



```
# Get the best lambda, i.e. the one with smallest MSE.
bestlam=cv.out$lambda.min
bestlam # = 473.0903
```

```
## [1] 326.0828
```

```
# Predict the test reposnse using the best lambda value
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])

# Compute the mean RSS. Much lower than the times we computed it above.
mean((ridge.pred-y.test)^2) # = 138602
```

```
## [1] 139976.2
```

```
# Look at the coefficients for the model with best lambda penelty.
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlam)[,1]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##  27.82233132   0.07799448   0.86129686   0.61951343   1.07116509   0.88281203
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##   1.62188685   1.36350867   0.01133608   0.05741783   0.40746636   0.11463333
##         CRBI       CWalks      LeagueA      LeagueN    DivisionW      PutOuts
##   0.12107007   0.05284627 -15.43723765  15.44047884 -78.98486531   0.16576003
##      Assists       Errors   NewLeagueN
##   0.02912505  -1.39032518   4.86177416
```