Assignment 3 Resampling

Dave Anderson

February 18, 2019

5.8 6.2 6.10

5.8)

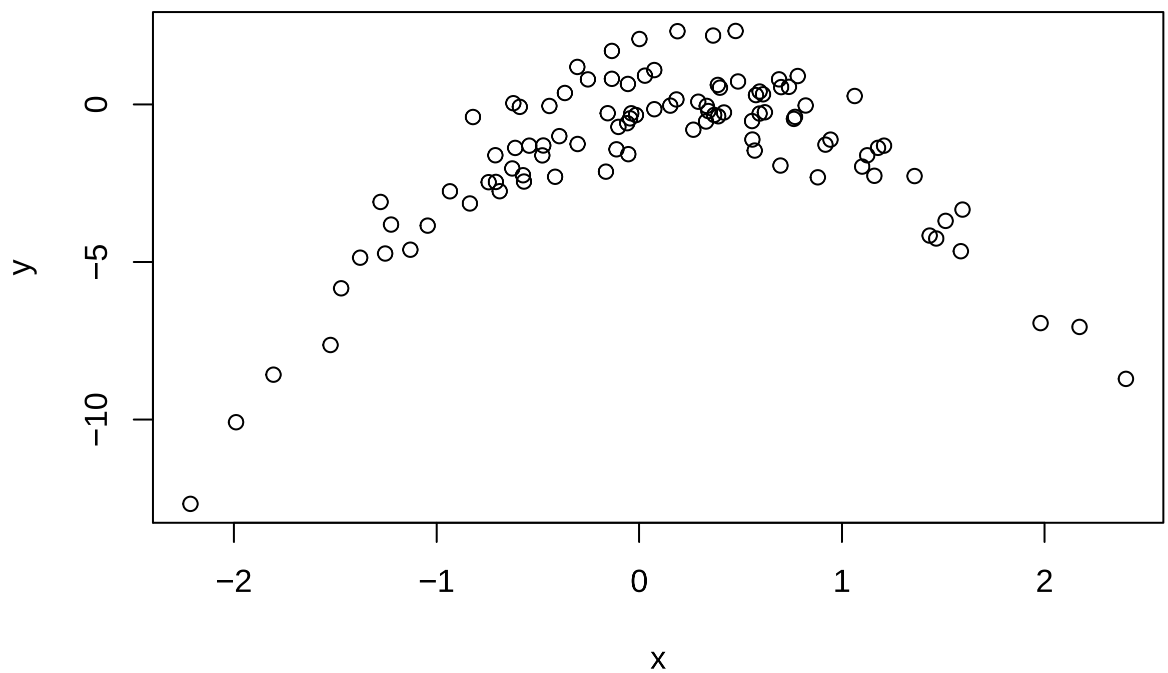
a)

```
set.seed(1)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
```

n = 100, p = 2 $Y = X - 2X^2 + e$

b)

```
plot(x,y)
```



The relationships between x and y is quadratic. x from -2 to 2, y from about -10 to 1

c)

i)

```
library(boot)
data = data.frame(x,y)
set.seed(1)
glm1 <- glm(y ~ x)
cv.glm(data,glm1)$delta
```

```
## [1] 7.288162 7.284744
```

ii)

```
glm2 <- glm(y ~poly(x,2))
cv.glm(data,glm2)$delta
```

```
## [1] 0.9374236 0.9371789
```

iii)

```
glm3 <- glm(y ~ poly(x,3))
cv.glm(data,glm3)$delta
```

```
## [1] 0.9566218 0.9562538
```

iv)

```
glm4 <- glm(y ~ poly(x,4))
cv.glm(data,glm4)$delta
```

```
## [1] 0.9539049 0.9534453
```

d)

i)

```
set.seed(2)
glm1 <- glm(y ~ x)
cv.glm(data,glm1)$delta
```

```
## [1] 7.288162 7.284744
```

ii)

```
glm2 <- glm(y ~poly(x,2))
cv.glm(data,glm2)$delta
```

```
## [1] 0.9374236 0.9371789
```

iii)

```
glm3 <- glm(y ~ poly(x,3))
cv.glm(data,glm3)$delta
```

```
## [1] 0.9566218 0.9562538
```

iv)

```
glm4 <- glm(y ~ poly(x,4))
cv.glm(data,glm4)$delta
```

```
## [1] 0.9539049 0.9534453
```

The results are the exact same, as expected, because LOOCV is evaluating the same n folds

e)

The quadratic model had the lowest test error rate, which is expected because we know the true nature of x and y is quadratic

f)

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = y ~ x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x             0.6925     0.2909   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(glm2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(glm3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, 3)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, 3)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, 3)3   0.26411    0.96263   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(glm4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

In each model, the linear and quadratic coefficients are the only significant estimates. This is similar to our CV results, but I didn't expect the linear coefficient to be significant

6.2)

a)

iii Lasso is less flexible and will provide more accurate predictions because of less variance and more bias. ##b) *iii Ridge regression produces similar results to lasso* ##c) *ii non-linear methods are more flexible, with less bias and more variance*

6.10)

a)

```
set.seed(2019)
p <- 20
n <- 1000
x <- matrix(rnorm(n * p),n,p)
B <- rnorm(p)
B[3] <- 0
B[5] <- 0
B[8] <- 0
B[11] <- 0
B[19] <- 0
e <- rnorm(p)
y <- x %*% B + e
```

b)

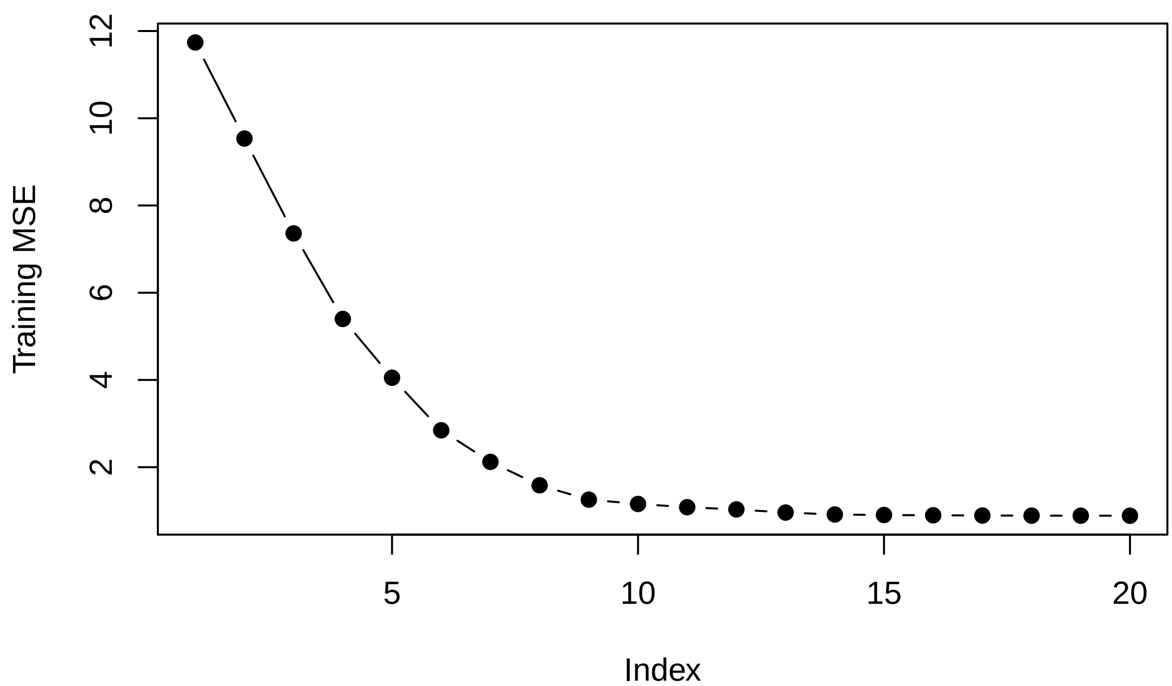
```
train <- sample(seq(1000),100,replace = FALSE)
y.train <- y[train,]
y.test <- y[-train,]
x.train <- x[train,]
x.test <- x[-train,]
```

c)

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.5.2
```

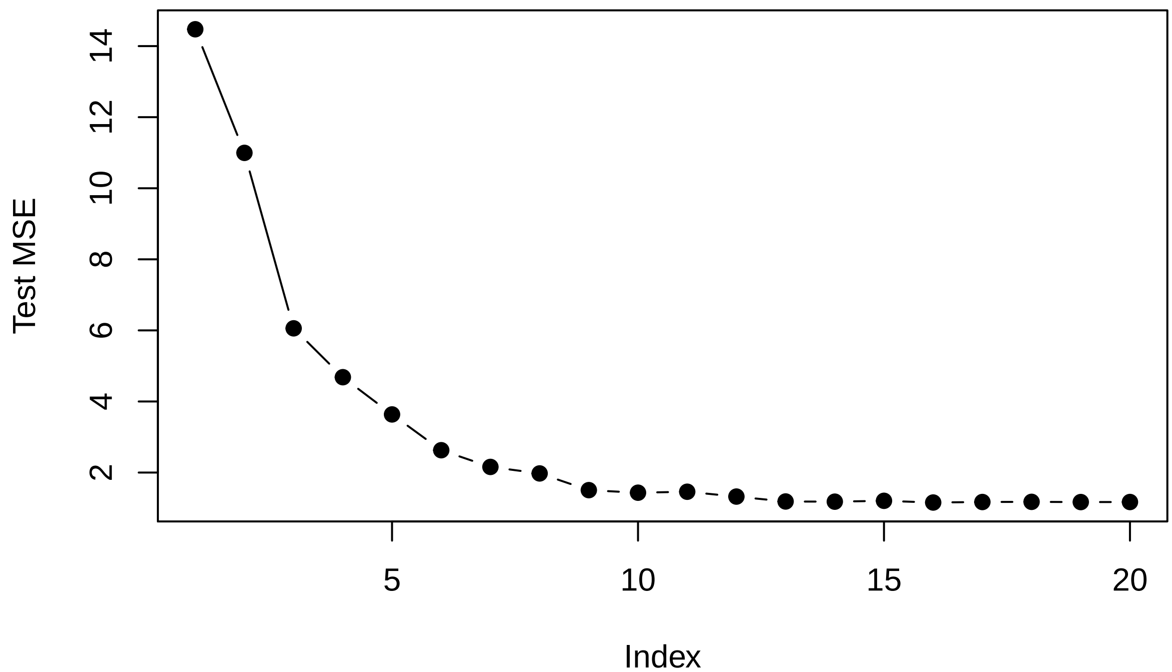
```
regfit.full <- regsubsets(y ~ ., data = data.frame(x = x.train, y = y.train),nvmax = p )
val.errors <- rep(NA,p)
x_cols <- colnames(x,do.NULL = FALSE,prefix = "x.")
for (i in 1:p) {
  coefi <- coef(regfit.full, id = i)
  pred <- as.matrix(x.train[,x_cols %in% names(coefi)]) %*% coefi[names(coefi) %in% x_cols]
  val.errors[i] <- mean((y.train - pred)^2)
}
plot(val.errors, ylab = "Training MSE", pch = 19, type = "b")
```



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)
```

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



e)

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

```
## [1] 16
```

The model with 14 predictors had the smallest test MSE

f)

```r
coef(regfit.full,id = 14)
```

```
## (Intercept)         x.1         x.2         x.4         x.7         x.9 
## -0.03827886  0.89787100  0.76970223 -1.80687018 -0.27733370  0.71245850 
##        x.10        x.12        x.13        x.14        x.15        x.16 
## -1.37370919  0.35077215 -1.94613332 -1.43911622 -0.78780829 -0.22461659 
##        x.17        x.18        x.20 
##  0.23775154 -0.26438887 -0.93993000
```

```r
B
```

```
##  [1]  0.8725894  0.5990908  0.0000000 -1.9256263  0.0000000 -0.2392388
```

```
##  [7] -0.2379375  0.0000000  0.6718665 -1.2905408  0.0000000  0.2941243
## [13] -1.9265471 -1.4018907 -0.6323060 -0.1135080  0.1164397 -0.4180595
## [19]  0.0000000 -0.9774071
```

The B coefficients used to generate the data that were set to zero (3,5,8,11,19) are all discluded from the variable selection. The other estimates are very close to what the true values are. x.6 was the only other variable not included, which had the smallest coefficient other than 0

g)