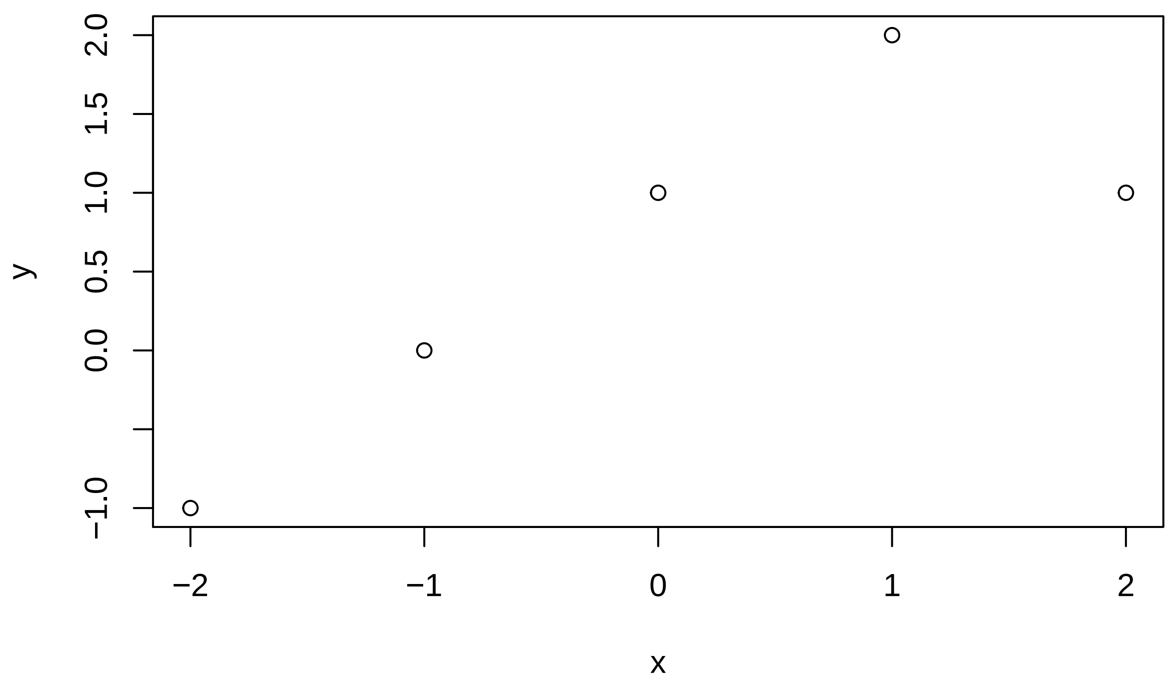
ISLR-HW5

Xiang XU

2/28/2019

##3

```r
x = -2:2
y = 1+x-2*(x-1)^2*I(x>1)
plot(x,y)
```



##9

```r
library(MASS)
data(Boston)

#a)
reg = lm(nox~poly(dis,3),data=Boston)
summary(reg)
```

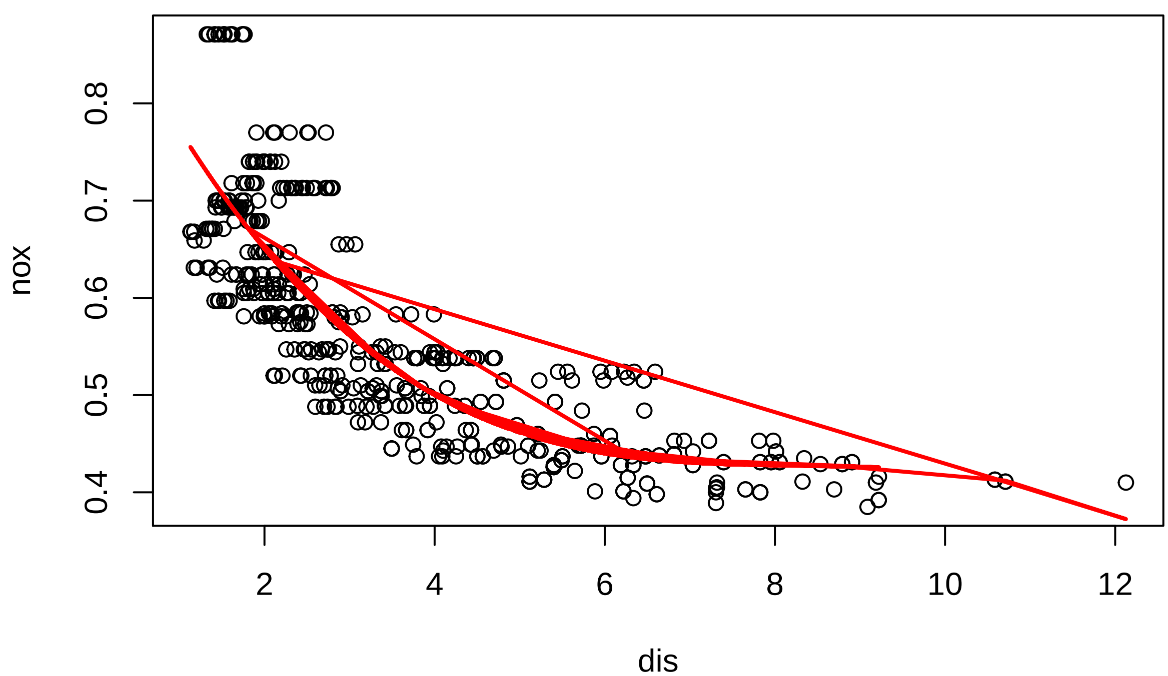
```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
coef(reg)
```

```
##   (Intercept) poly(dis, 3)1 poly(dis, 3)2 poly(dis, 3)3
##     0.5546951    -2.0030959     0.8563300    -0.3180490
```

```r
reg.pred <- predict(reg,dis=list(Boston$dis))
plot(nox~dis,data=Boston)
lines(Boston$dis,reg.pred,col="red",lwd=2)
```

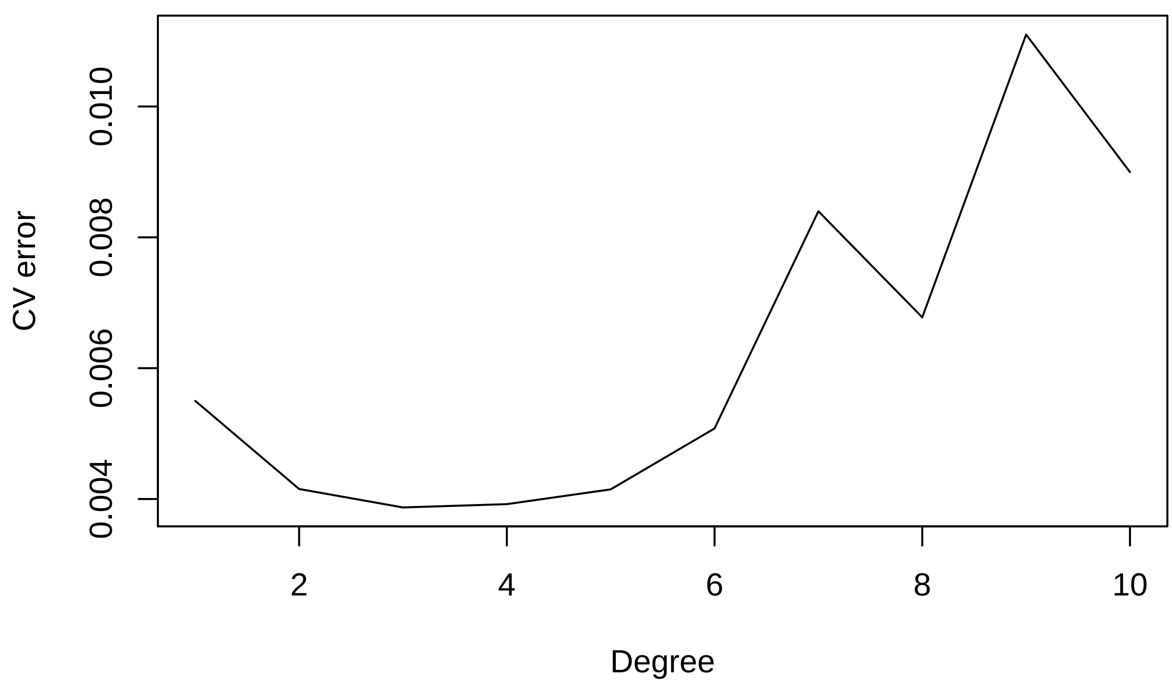



```r
#b)
rss <- rep(NA,10)
for (i in 1:10){
  reg <- lm(nox~poly(dis,i),data=Boston)
```

```
  rss[i] <- sum(reg$residuals^2)
}
rss
```

```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484
##  [8] 1.835630 1.833331 1.832171
```

```
# We can see that as i increases, the RSS decreases.

#c
library(boot)
error <- rep(NA,10)
for (i in 1:10){
  reg <- glm(nox~poly(dis,i),data=Boston)
  error[i] <- cv.glm(Boston,reg,K=10)$delta[2]
}
plot(1:10,error,xlab="Degree",ylab="CV error",type="line")
```

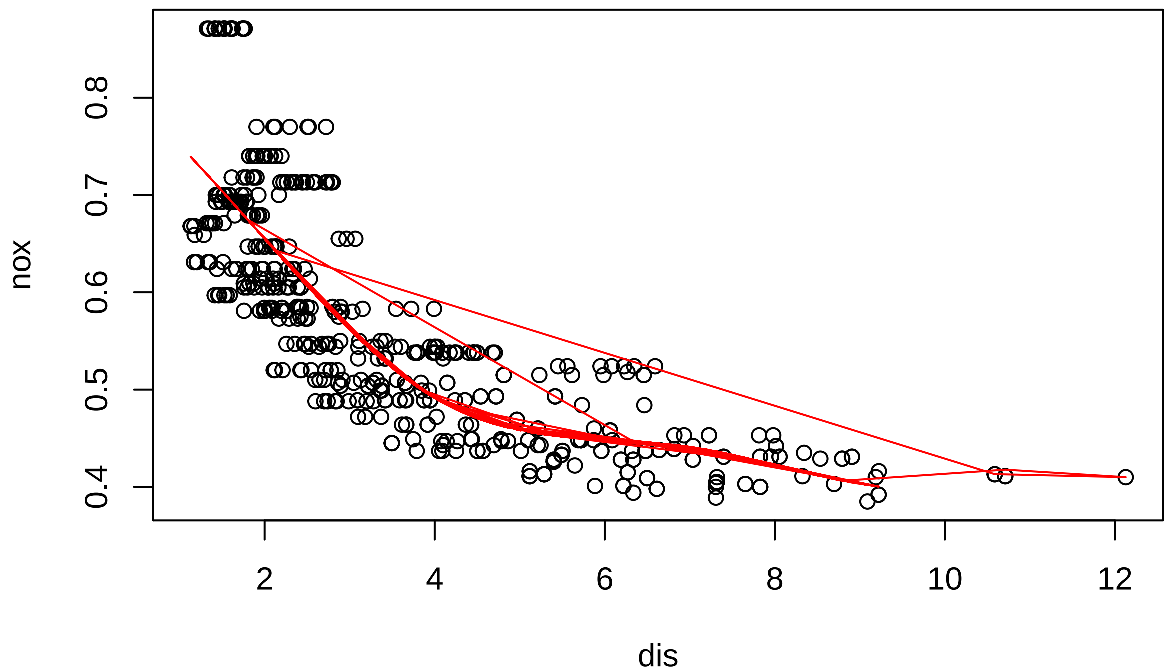



```
#From the plot, we may want to select degree of three.

#d
library(splines)
reg <- lm(nox~bs(dis,df=4,knots=c(4,7,11)),data=Boston)
summary(reg)
```

```
##
## Call:
```

```
## lm(formula = nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124567 -0.040355 -0.008702  0.024740  0.192920
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              0.73926    0.01331  55.537  < 2e-16
## bs(dis, df = 4, knots = c(4, 7, 11))1 -0.08861    0.02504  -3.539  0.00044
## bs(dis, df = 4, knots = c(4, 7, 11))2 -0.31341    0.01680 -18.658  < 2e-16
## bs(dis, df = 4, knots = c(4, 7, 11))3 -0.26618    0.03147  -8.459 3.00e-16
## bs(dis, df = 4, knots = c(4, 7, 11))4 -0.39802    0.04647  -8.565  < 2e-16
## bs(dis, df = 4, knots = c(4, 7, 11))5 -0.25681    0.09001  -2.853  0.00451
## bs(dis, df = 4, knots = c(4, 7, 11))6 -0.32926    0.06327  -5.204 2.85e-07
##
## (Intercept)                           ***
## bs(dis, df = 4, knots = c(4, 7, 11))1 ***
## bs(dis, df = 4, knots = c(4, 7, 11))2 ***
## bs(dis, df = 4, knots = c(4, 7, 11))3 ***
## bs(dis, df = 4, knots = c(4, 7, 11))4 ***
## bs(dis, df = 4, knots = c(4, 7, 11))5 **
## bs(dis, df = 4, knots = c(4, 7, 11))6 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06185 on 499 degrees of freedom
## Multiple R-squared:  0.7185, Adjusted R-squared:  0.7151
## F-statistic: 212.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

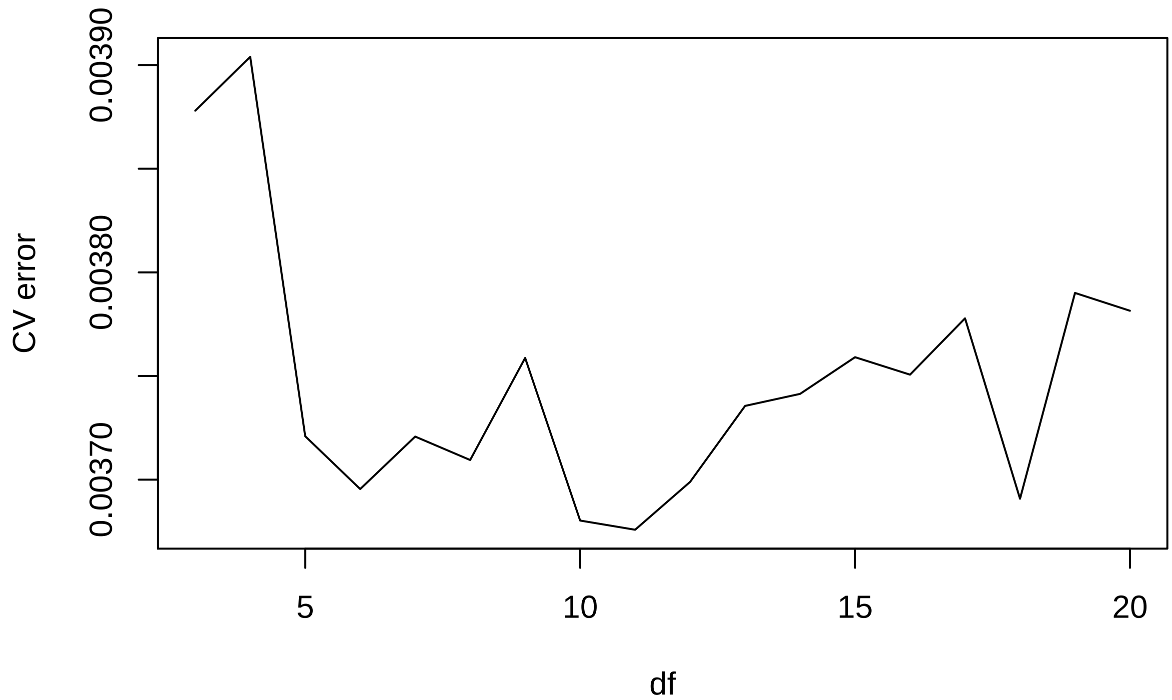
```r
pred <- predict(reg,dis=list(Boston$dis))
plot(nox~dis,data=Boston)
lines(Boston$dis,pred,col="red")
```




```r
#e
rss <- rep(NA,17)
for (i in 3:20){ #df should be greater than three
  reg <- lm(nox~bs(dis,df=i),data=Boston)
  rss[i] <- sum(reg$residuals^2)
}
rss[3:20]
```

```
##  [1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653
##  [8] 1.792535 1.796992 1.788999 1.782350 1.781838 1.782798 1.783546
## [15] 1.779789 1.775838 1.774487 1.776727
```

```r
#f
error <- rep(NA,20)
for (i in 3:20){
  reg <- glm(nox~bs(dis,df=i),data=Boston)
  error[i] <- cv.glm(Boston,reg,K=10)$delta[2]
}
plot(3:20,error[3:20],xlab="df",ylab="CV error",type="l")
```



##10

```
#a
set.seed(1)
library(ISLR)
```

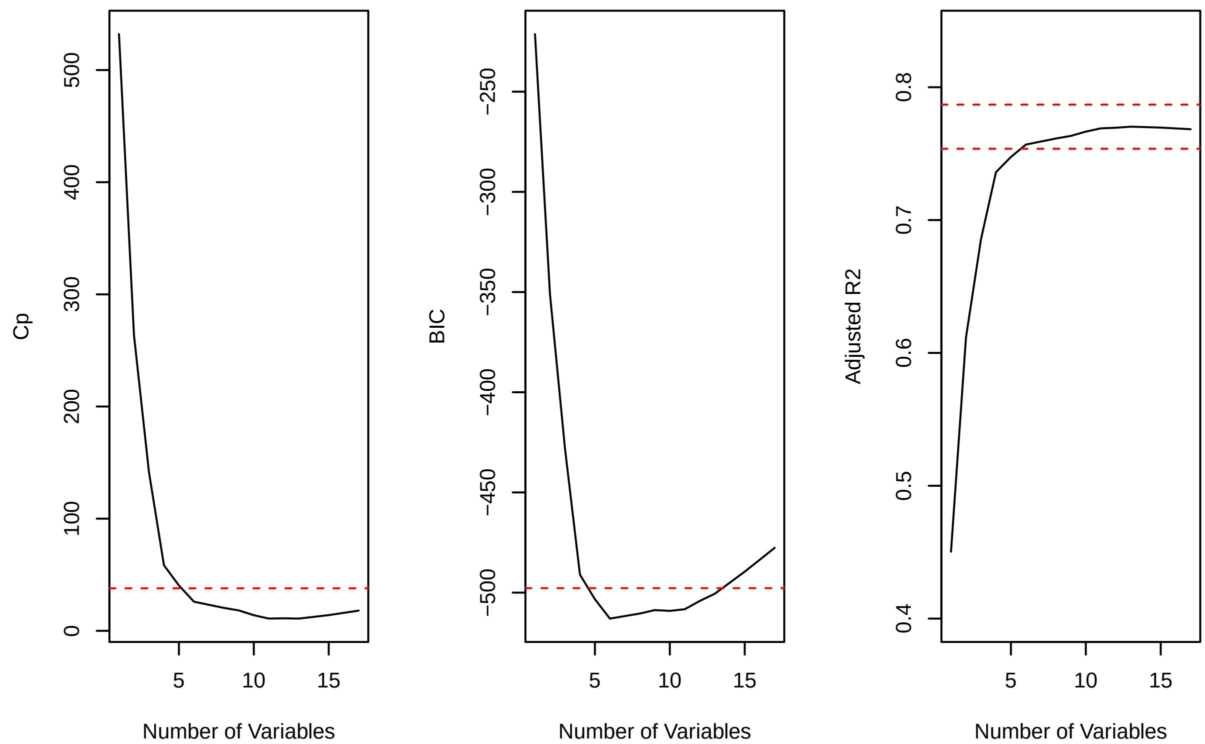
```
## Warning: package 'ISLR' was built under R version 3.5.2
```

```
library(leaps)
```

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

```
data("College")
train <- sample(length(College$Outstate),length(College$Outstate)/2)
Col.train <- College[train,]
Col.test <- College[-train,]
reg <- regsubsets(Outstate~.,data=Col.train,nvmax=17,method = "forward")
reg.sum <- summary(reg)
par(mfrow=c(1,3))
plot(reg.sum$cp,xlab="Number of Variables",ylab="Cp",type="l")
min.cp = min(reg.sum$cp)
std.cp = sd(reg.sum$cp)
abline(h = min.cp + 0.2 * std.cp, col = "red", lty = 2)
abline(h = min.cp - 0.2 * std.cp, col = "red", lty = 2)
plot(reg.sum$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
min.bic = min(reg.sum$bic)
std.bic = sd(reg.sum$bic)
abline(h = min.bic + 0.2 * std.bic, col = "red", lty = 2)
```

```r
abline(h = min.bic - 0.2 * std.bic, col = "red", lty = 2)
plot(reg.sum$adjr2, xlab = "Number of Variables", ylab = "Adjusted R2",
    type = "l", ylim = c(0.4, 0.84))
max.adjr2 = max(reg.sum$adjr2)
std.adjr2 = sd(reg.sum$adjr2)
abline(h = max.adjr2 + 0.2 * std.adjr2, col = "red", lty = 2)
abline(h = max.adjr2 - 0.2 * std.adjr2, col = "red", lty = 2)
```



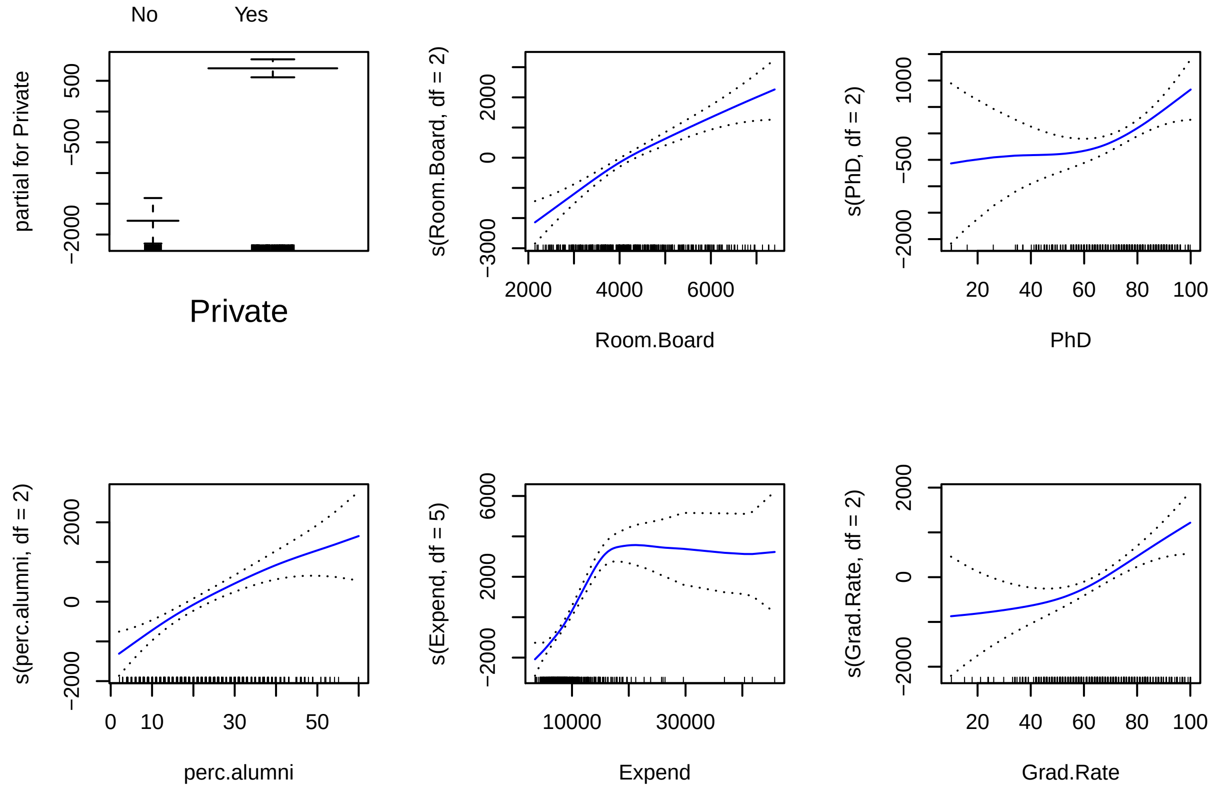

```r
#From these plot, we may want to select 6 as the best subset size.

#b
library(gam)
```

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

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

```
## Loaded gam 1.16
```

```r
reg <- gam(Outstate~Private+s(Room.Board,df=2)+s(PhD,df=2)+
             s(perc.alumni, df = 2) + s(Expend, df = 5) +
             s(Grad.Rate, df = 2), data = Col.train)
par(mfrow=c(2,3))
plot(reg,se=T,col="blue")
```




```r
#c
pred <- predict(reg,Col.test)
error <- mean((Col.test$Outstate-pred)^2)
error
```

```
## [1] 3745460
```

```r
tss = mean((Col.test$Outstate - mean(Col.test$Outstate))^2)
test.rss = 1 - error/tss
test.rss
```

```
## [1] 0.7696916
```

```r
#Using 6 variables, we got a R-square of 77%.

#d
summary(reg)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + s(PhD,
##     df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate,
##     df = 2), data = Col.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -4977.74 -1184.52    58.33  1220.04  7688.30
##
## (Dispersion Parameter for gaussian family taken to be 3300711)
##
```

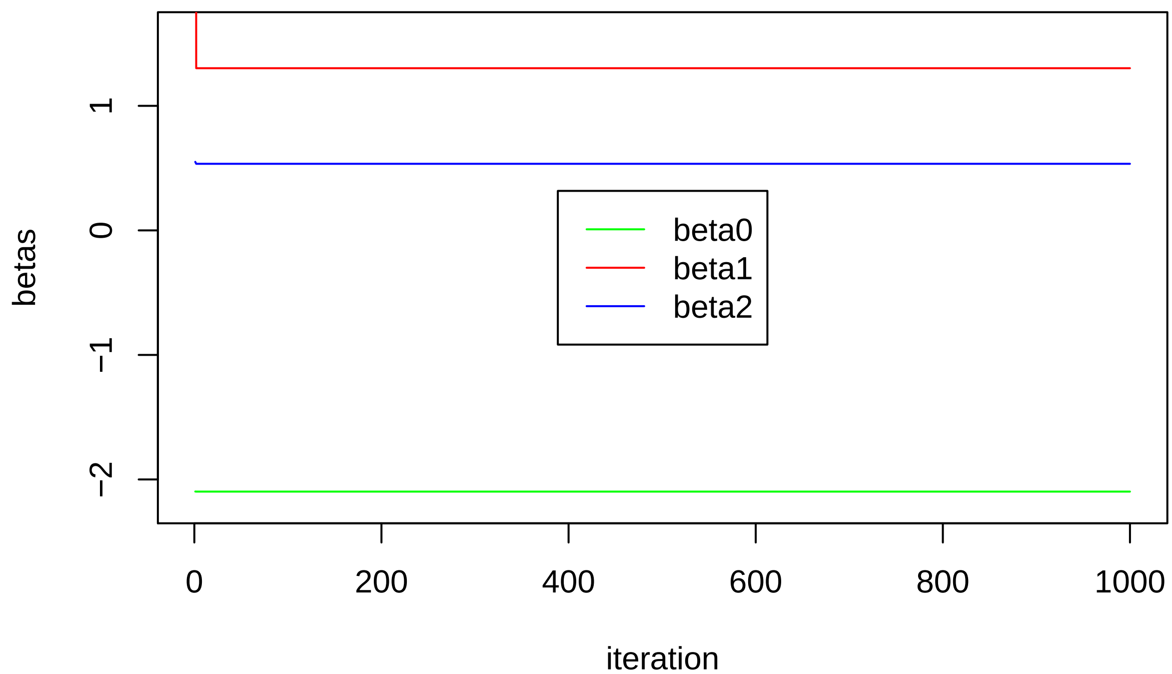
```
##     Null Deviance: 6221998532 on 387 degrees of freedom
## Residual Deviance: 1231165118 on 373 degrees of freedom
## AIC: 6941.542
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                           Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private                    1 1779433688 1779433688 539.106 < 2.2e-16 ***
## s(Room.Board, df = 2)      1 1221825562 1221825562 370.171 < 2.2e-16 ***
## s(PhD, df = 2)             1  382472137  382472137 115.876 < 2.2e-16 ***
## s(perc.alumni, df = 2)     1  328493313  328493313  99.522 < 2.2e-16 ***
## s(Expend, df = 5)          1  416585875  416585875 126.211 < 2.2e-16 ***
## s(Grad.Rate, df = 2)       1   55284580   55284580  16.749 5.232e-05 ***
## Residuals                373 1231165118    3300711
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                         Npar Df  Npar F     Pr(F)
## (Intercept)
## Private
## s(Room.Board, df = 2)        1  3.5562   0.06010 .
## s(PhD, df = 2)               1  4.3421   0.03786 *
## s(perc.alumni, df = 2)       1  1.9158   0.16715
## s(Expend, df = 5)            4 16.8636 1.016e-12 ***
## s(Grad.Rate, df = 2)         1  3.7208   0.05450 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#From the Nonparametric Effects' Anova:
#A strong evidence of non-linear relationship between response variable and expend.
```

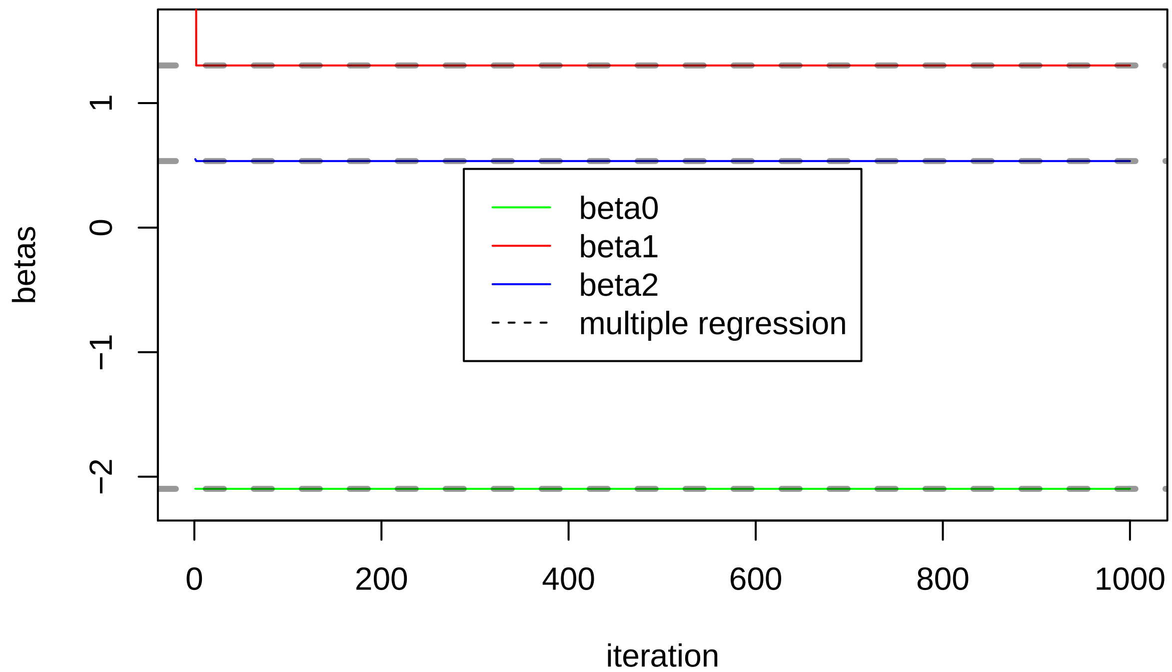
#11

```r
#a
set.seed(1)
X1 = rnorm(100)
X2 = rnorm(100)
eps = rnorm(100, sd = 0.1)
Y = -2.1 + 1.3 * X1 + 0.54 * X2 + eps

#b
beta0 = rep(NA, 1000)
beta1 = rep(NA, 1000)
beta2 = rep(NA, 1000)
beta1[1] = 18

#c
for (i in 1:1000) {
    a = Y - beta1[i] * X1
    beta2[i] = lm(a ~ X2)$coef[2]
    a = Y - beta2[i] * X2
    lm.fit = lm(a ~ X1)
    if (i < 1000) {
```

```r
        beta1[i + 1] = lm.fit$coef[2]
    }
    beta0[i] = lm.fit$coef[1]
}
plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-2.2,
    1.6), col = "green")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")
legend("center", c("beta0", "beta1", "beta2"), lty = 1, col = c("green", "red",
    "blue"))
```




```r
#Notice that the coefficients quickly attain stable points.

#f
lm.fit = lm(Y ~ X1 + X2)
plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-2.2,
    1.6), col = "green")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")
abline(h = lm.fit$coef[1], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))
abline(h = lm.fit$coef[2], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))
abline(h = lm.fit$coef[3], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))
legend("center", c("beta0", "beta1", "beta2", "multiple regression"), lty = c(1,
    1, 1, 2), col = c("green", "red", "blue", "black"))
```




```
#g
# We only need one iteration to obtain a good approximation.
```