HW6

Ramtin Boustani - SUID# 05999261

Problem 1

Chapter 7, Exercise 2

(a)

$\hat{g} = 0$ (zero)

(b)

$\hat{g} = c$ (constant)
First derivative measures the slope and larger lambda causes get close to constant.

(c)

$\hat{g} = ax + b$ (linear)
Second derivative peaks wiggle of the function also the penalty term captures all non-linearly in the function. Smallest lambda more wiggly function and larger lambda causes more linear function

(d)

$\hat{g} = ax^2 + bx + c$ (cubic)

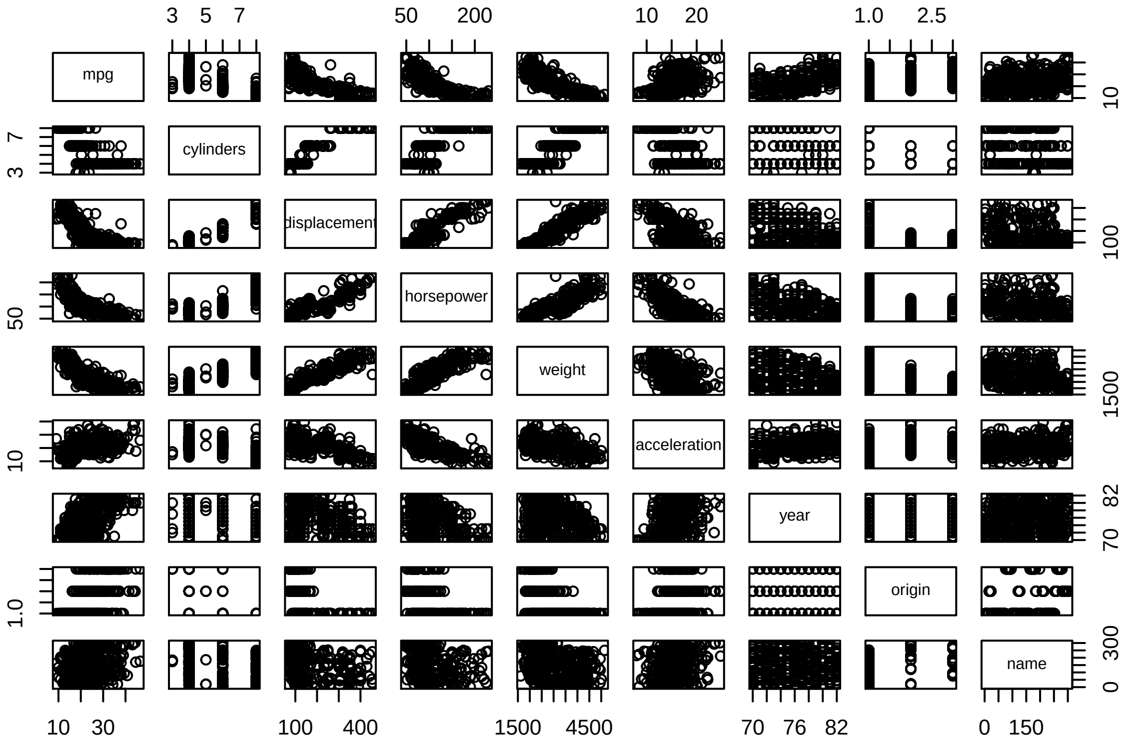
(e)

Simple minimizing RSS. There is no penality so it is interpolating spline. (Overfit data and too flexible)

Problem 2

Chapter 7, Exercise 8

```
require(ISLR)
require(boot)
require(splines)
require(gam)
attach(Auto)
set.seed(1)
pairs(Auto)
```



mpg relationship with displacement

Polynomial

Using 10 folds cross validations

```
set.seed(1)
errors = rep(NA, 15)
for (d in 1:15){
  fit = glm(mpg ~poly(displacement, d), data=Auto)
  errors[d] = cv.glm(Auto, fit, K=10)$delta[2]
}
which.min(errors)
```

```
## [1] 10
```

```
errors[which.min(errors)]
```

```
## [1] 17.4852
```

Polynomial degree 10th with test error 17.48

Fixed knots regression Splines

```
set.seed(1)
errors = rep(NA, 10)
for (d in 2:10){
  fit = glm(mpg ~ ns(displacement, df=d), data = Auto)
  errors[d] = cv.glm(Auto, fit, K=10)$delta[2]
}
which.min(errors)
```

```
## [1] 9
```

```r
errors[which.min(errors)]
```

```
## [1] 17.61242
```

Splines with 9 degree of freedom with test error 17.61

Linear regression

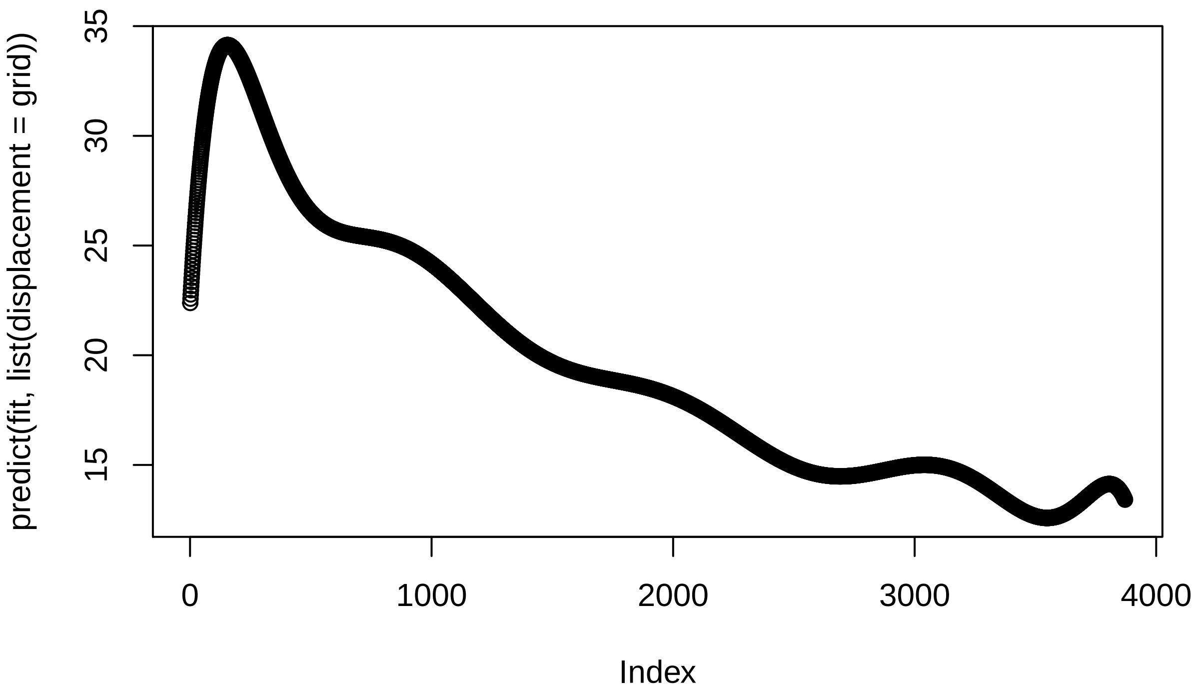
```r
glm.fit = glm(mpg ~ displacement, data = Auto)
cv.glm(glm.fit,data = Auto)$delta[2]
```

```
## [1] 21.59218
```

Conclusion

Comparing mpg ~ displacement
Linear: 21.59218
Polynomial(degree 10th): 17.48
Splines(df=9): 17.61
Shows nonlinear modeles have lower test error!

```r
fit = glm(mpg ~poly(displacement, 10), data=Auto)
grid = seq(range(displacement)[1],range(displacement)[2], 0.1)
plot(predict(fit, list(displacement =  grid )))
```



mpg relationship with weight

Polynomial

Using 10 folds cross validations

```r
set.seed(1)
errors = rep(NA, 15)
for (d in 1:15){
  fit = glm(mpg ~poly(weight, d), data=Auto)
```

```
  errors[d] = cv.glm(Auto, fit, K=10)$delta[2]
}
which.min(errors)
```

```
## [1] 2
```

```
errors[which.min(errors)]
```

```
## [1] 17.55181
```

Quadratic with test error 17.55

Fixed knots regression Splines

```
set.seed(1)
errors = rep(NA, 10)
for (d in 2:10){
  fit = glm(mpg ~ ns(weight, df=d), data = Auto)
  errors[d] = cv.glm(Auto, fit, K=10)$delta[2]
}
which.min(errors)
```

```
## [1] 10
```

```
errors[which.min(errors)]
```

```
## [1] 17.37649
```

Splines with 10 degree of freedom with test error 17.37

Linear regression

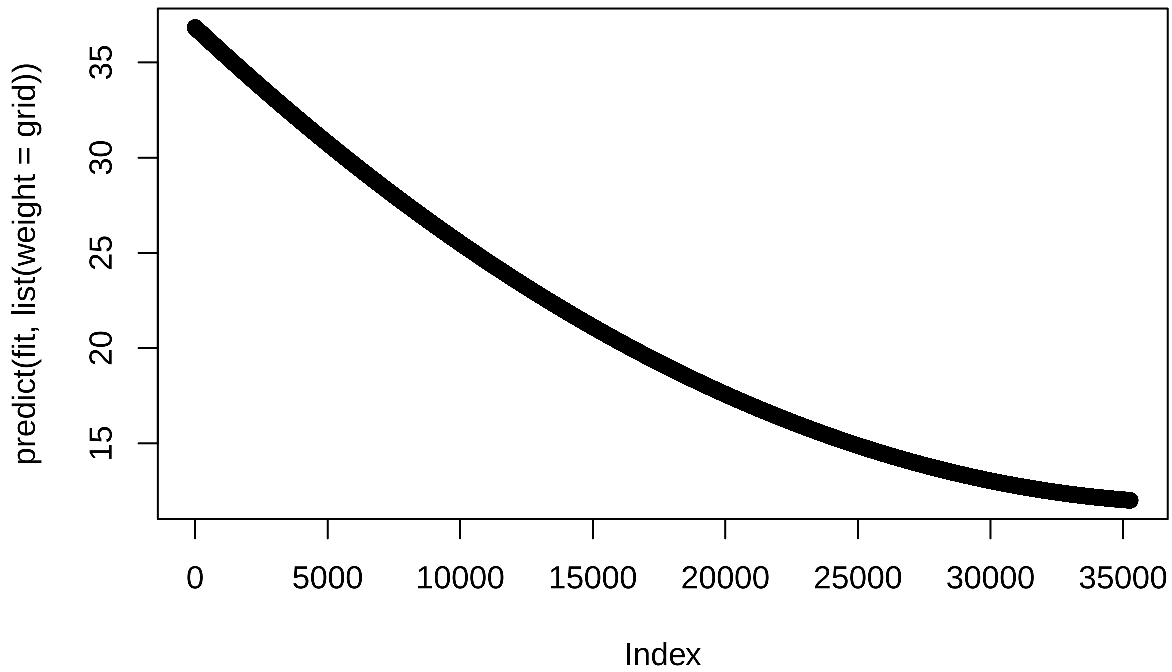
```
glm.fit = glm(mpg ~ weight, data = Auto)
cv.glm(glm.fit,data = Auto)$delta[2]
```

```
## [1] 18.85139
```

Conclusion

Comparing mpg ~ weight
Linear: 18.85
Polynomial(degree 2th): 17.55
Splines(df=10): 17.37
Nonlinear slightly better than linear model

```
fit = glm(mpg ~poly(weight, 2), data=Auto)
grid = seq(range(weight)[1],range(weight)[2], 0.1)
plot(predict(fit, list(weight =  grid )))
```



Problem 3

Chapter 7, Exercise 9

```r
require(MASS)
```

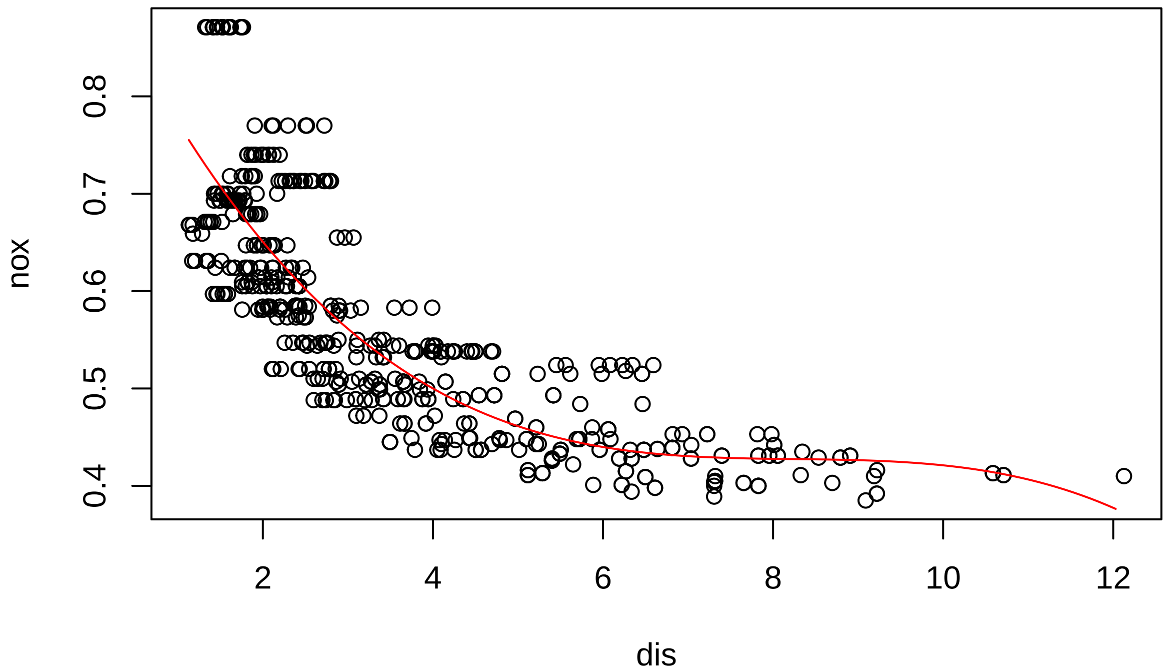
```
## Loading required package: MASS
```

```r
attach(Boston)
```

(a)

Cubic polynomial regression

```r
lm.fit = glm(nox~poly(dis, 3), data=Boston)
dis.grid = seq(from=Boston$dis[which.min(Boston$dis)], to=Boston$dis[which.max(Boston$dis)], by=0.1)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(y=nox, x=dis)
lines(dis.grid, lm.pred, col="red")
```

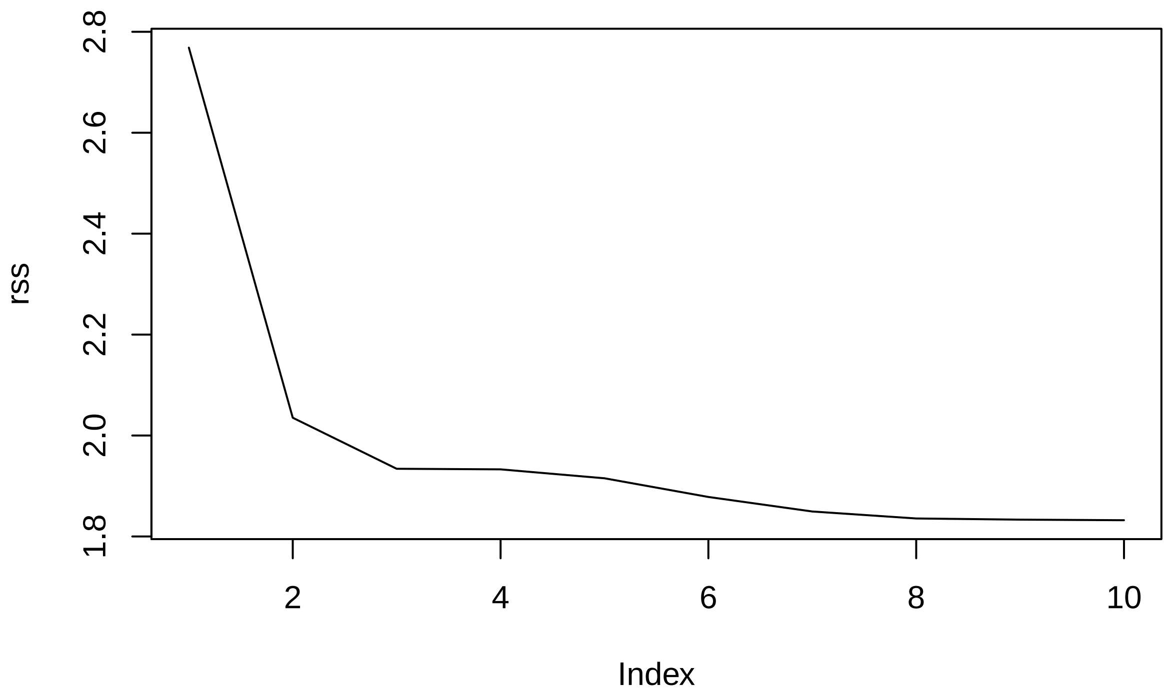


(b)

Training RSS from degree 1 to 10

```
rss = rep(NA, 10)
for (d in 1:10){
  fit = glm(nox~poly(dis, d), data=Boston)
  rss[d] = sum(fit$residuals^2)
}
plot(rss, type="line")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
## first character
```



Training error RSS monotonically decrease with increasing degree of polynomial

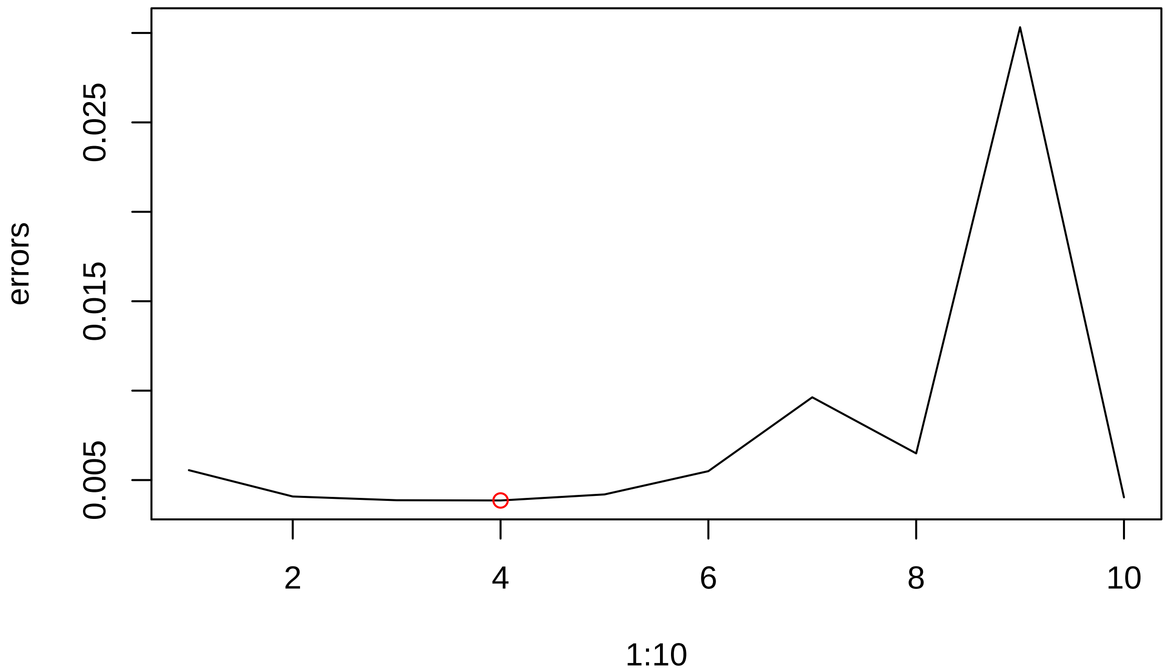
(c)

Cross-validation

```
set.seed(1)
errors = rep(NA, 10)
for (d in 1:10){
  fit = glm(nox ~poly(dis, d), data=Boston)
  errors[d] = cv.glm(Boston, fit, K=10)$delta[2]
}
plot(1:10, errors, type="line")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
## first character
```

```
points(x=which.min(errors), y=errors[which.min(errors)], col="red")
```



Based on CV test error, 4 is a good ploynomial degree

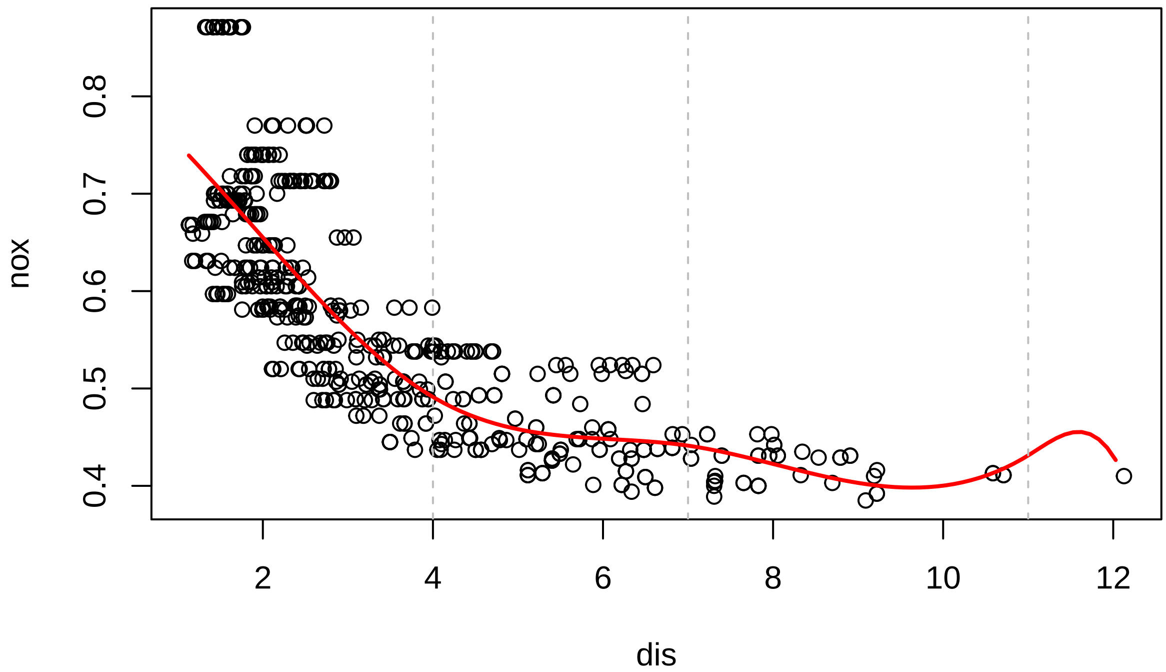
(d)

```
range(dis)
```

```
## [1]  1.1296 12.1265
```

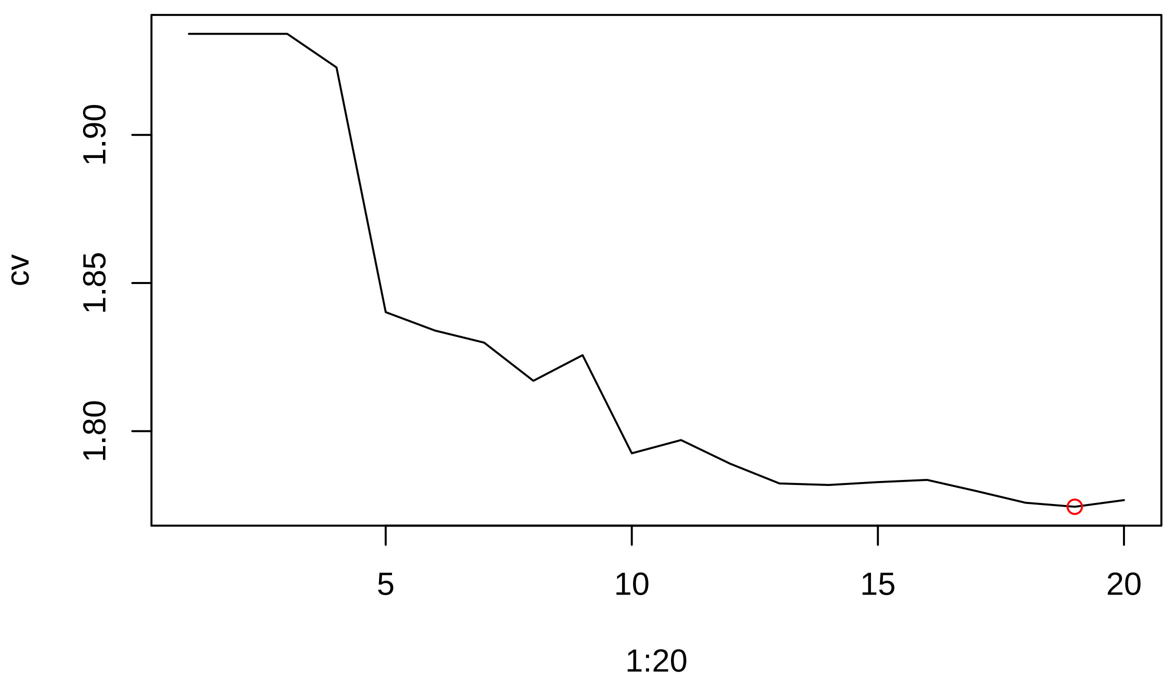
range between 1 to 13 splitting 4 equal intervals and providing as knots

```
lm.fit = glm(nox~bs(dis,knots = c(4,7,11)), data=Boston)
dis.grid = seq(from=Boston$dis[which.min(Boston$dis)], to=Boston$dis[which.max(Boston$dis)], by=0.1)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(y=nox, x=dis)
lines(dis.grid, lm.pred, col="red", lwd=2)
abline(v=4, col="grey", lty="dashed")
abline(v=7, col="grey", lty="dashed")
abline(v=11, col="grey", lty="dashed")
```



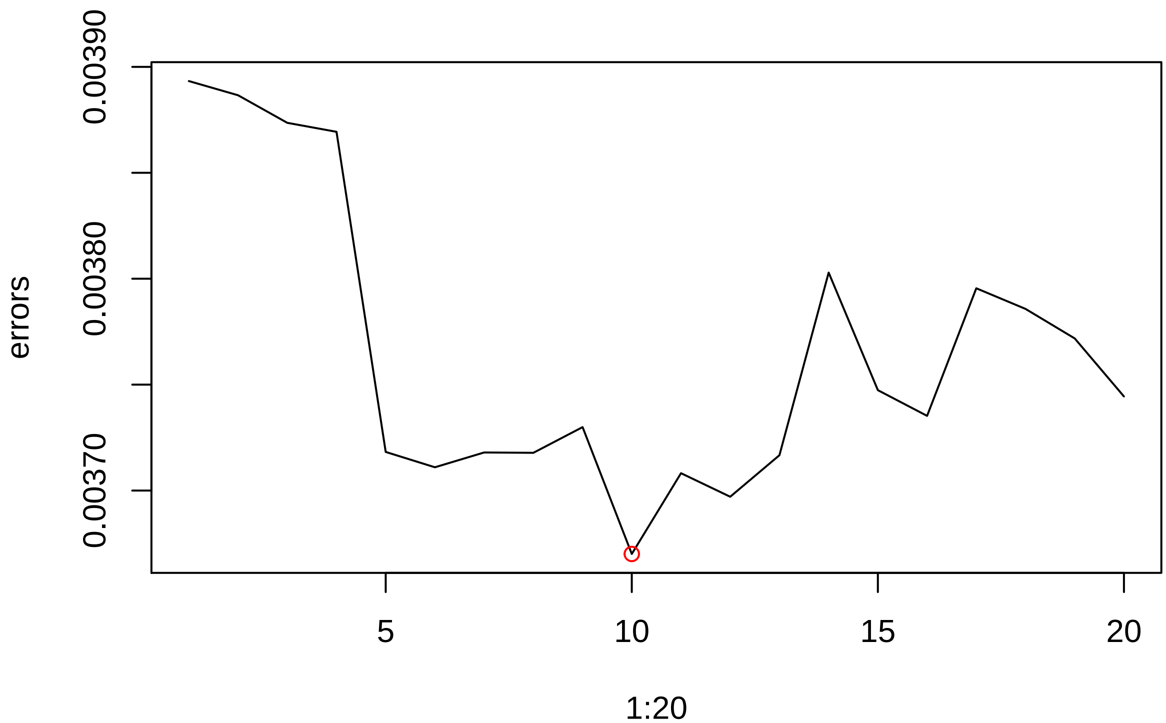
(e)

```
cv = rep(NA, 20)
for(d in 1:20){
  fit = lm(nox ~ bs(dis, df = d), data = Boston)
  cv[d] = sum((fit$residuals)^2)
}
plot(1:20, cv, type = "line")
points(x=which.min(cv), y=cv[which.min(cv)], col="red")
```



(f)

```
set.seed(1)
errors = rep(NA, 20)
for(d in 1:20){
  fit = glm(nox ~ bs(dis, df = d), data = Boston)
  errors[d] = cv.glm(data = Boston, glmfit = fit, K=10)$delta[2]
}
plot(1:20, errors, type = "line")
points(x=which.min(errors), y=errors[which.min(errors)], col="red")
```



10 is good degree of freedom for regression spline

Problem 4

Chapter 7, Exercise 10

```
require(leaps)
attach(College)
set.seed(1)
```

(a)

```
n = dim(College)[1]
train = sample(n, n/2, replace=FALSE)
reg.fit = regsubsets(Outstate~. ,data = College[train,], nvmax=17, method = "forward" )
reg.summary = summary(reg.fit)
par(mfrow = c(1, 2))
plot(reg.summary$adjr2, type = "line", main = "adjr2",xlab = "Number of Variables")
```

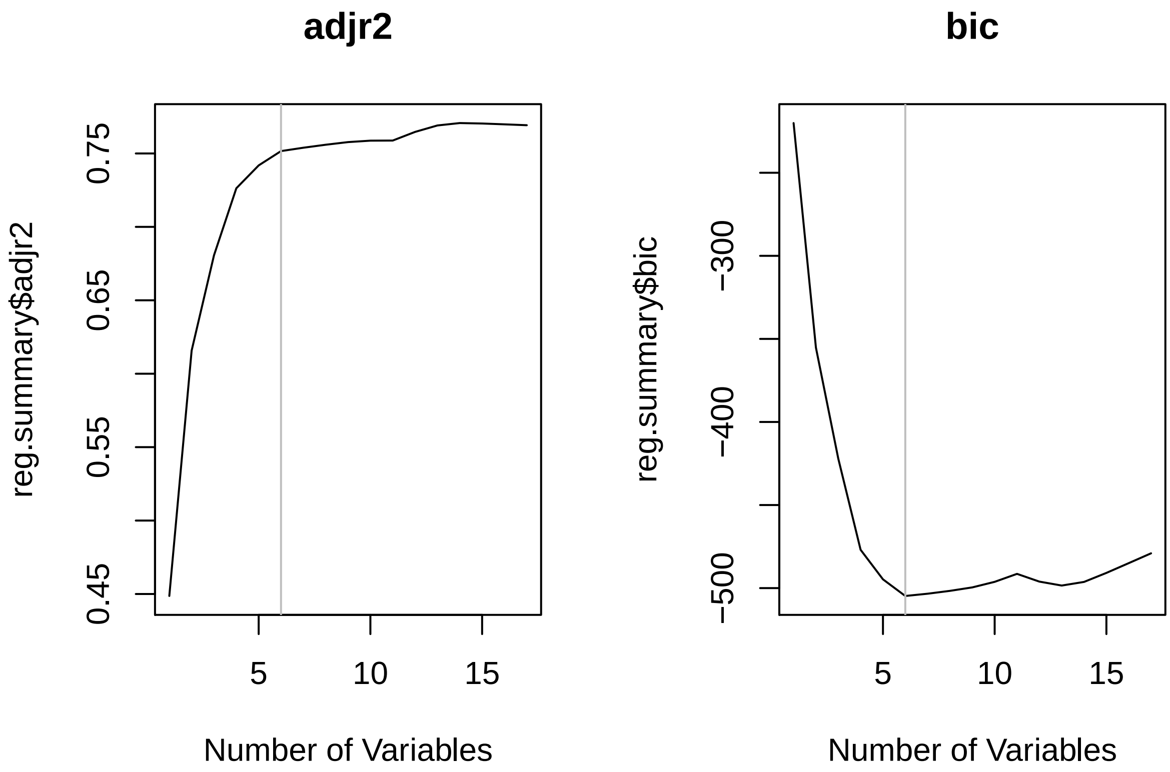
```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
```

```
## first character
```

```r
abline(v=6, col="grey")
plot(reg.summary$bic, type = "line", main="bic", xlab = "Number of Variables")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
## first character
```

```r
abline(v=6, col="grey")
```



6 is a good number of variables for this model, we train again with full data to find variable names

```r
reg.fit = regsubsets(Outstate ~ ., data = College, method = "forward")
names(coef(reg.fit, id=6))
```

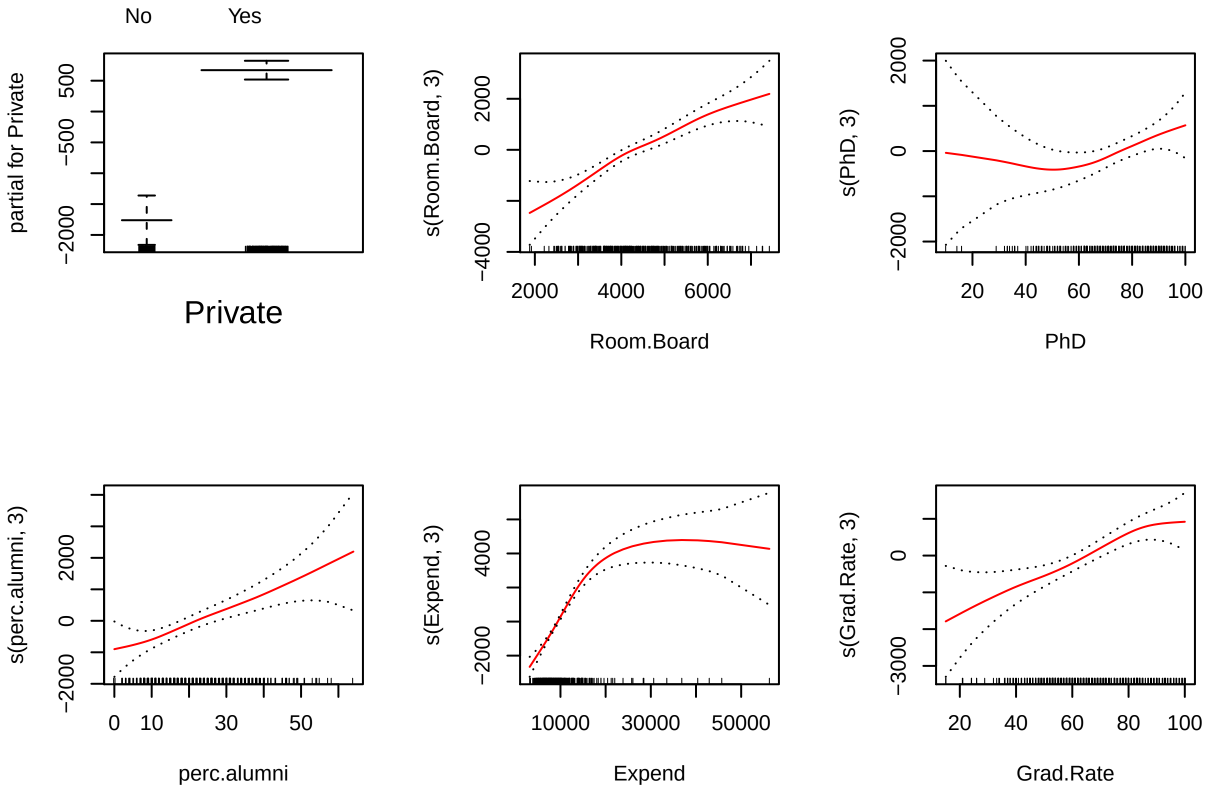
```
## [1] "(Intercept)" "PrivateYes"  "Room.Board"  "PhD"         "perc.alumni"
## [6] "Expend"      "Grad.Rate"
```

(b)

```r
set.seed(1)
gam.fit = gam(Outstate ~ Private + s(Room.Board, 3) + s(PhD, 3) + s(perc.alumni, 3) + s(Expend, 3) + s(
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts
## argument ignored
```

```r
par(mfrow=c(2,3))
plot(gam.fit, se = TRUE, col="red")
```



(c)

```
lm.fit = lm(Outstate ~Private+Room.Board+PhD+perc.alumni+Expend+Grad.Rate, data = College[train,])
lm.pred = predict(lm.fit, newdata = College[-train,])
mean((College[-train,]$Outstate - lm.pred)^2)
```

```
## [1] 3841483
```

```
set.seed(1)
gam.pred = predict(gam.fit, newdata = College[-train,])
mean((College[-train,]$Outstate - gam.pred)^2)
```

```
## [1] 3353709
```

Comparing linear list sqaure with GAM shows GAM has slightly lower RSS

(d)

```
summary(gam.fit)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, 3) + s(PhD,
##     3) + s(perc.alumni, 3) + s(Expend, 3) + s(Grad.Rate, 3),
##     data = College[train, ])
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -6963.2 -1131.7  -101.2  1322.2  7949.7
##
## (Dispersion Parameter for gaussian family taken to be 3821609)
##
```

```
##     Null Deviance: 6989966760 on 387 degrees of freedom
## Residual Deviance: 1417814885 on 370.9995 degrees of freedom
## AIC: 7000.312
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                      Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private               1 1767246309 1767246309 462.435 < 2.2e-16 ***
## s(Room.Board, 3)      1 1580386922 1580386922 413.540 < 2.2e-16 ***
## s(PhD, 3)             1  351828206  351828206  92.063 < 2.2e-16 ***
## s(perc.alumni, 3)     1  338018768  338018768  88.449 < 2.2e-16 ***
## s(Expend, 3)          1  498727240  498727240 130.502 < 2.2e-16 ***
## s(Grad.Rate, 3)       1   85973130   85973130  22.497 3.008e-06 ***
## Residuals           371 1417814885    3821609
## ---
## 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, 3)          2  1.6491    0.1936
## s(PhD, 3)                 2  1.2597    0.2850
## s(perc.alumni, 3)         2  0.2914    0.7474
## s(Expend, 3)              2 30.9997 3.55e-13 ***
## s(Grad.Rate, 3)           2  1.0910    0.3369
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Anova for Nonparametric Effects shows strong relation between OutState and Expend

Problem 5

Chapter 7, Exercise 11

(a)

```
set.seed(1)
X1 = rnorm(100,1)
X2 = rnorm(100,1)
eps = rnorm(100,0.1)
b0 = 3
b1 = 5
b2 = -1
Y = b0 + b1*X1 + b2*X2 + eps
```

(b)

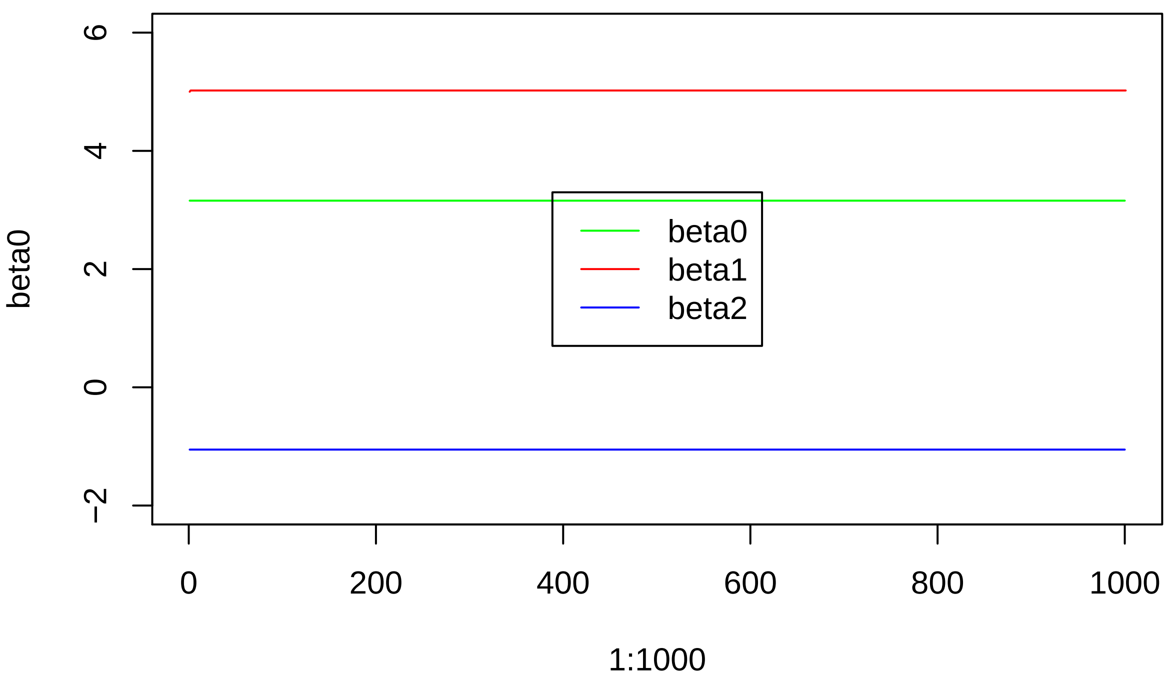
```
beta0 = rep(NA,1000)
beta1 = rep(NA,1000)
beta2 = rep(NA,1000)
```

(c) (d) (e)

```
beta1[1] = 5
for (i in 1:1000){
  #fixing beta1
  a = Y - beta1[i]*X1
  lm.fit = lm(a~X2)
  beta2[i] = lm.fit$coefficients[2]

  #fixing beta2
  a = Y - beta2[i]*X2
  lm.fit = lm(a~X1)
  beta1[i+1] = lm.fit$coefficients[2]

  beta0[i] = lm.fit$coefficients[1]
}

plot(1:1000, beta0, type = "line", col="green",  ylim = c(-2, 6))
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
## first character
```

```
lines(1:1001, beta1, col="red")
lines(1:1000, beta2, col="blue")
legend("center", c("beta0", "beta1", "beta2"), lty = 1, col = c("green", "red", "blue"))
```



(f)

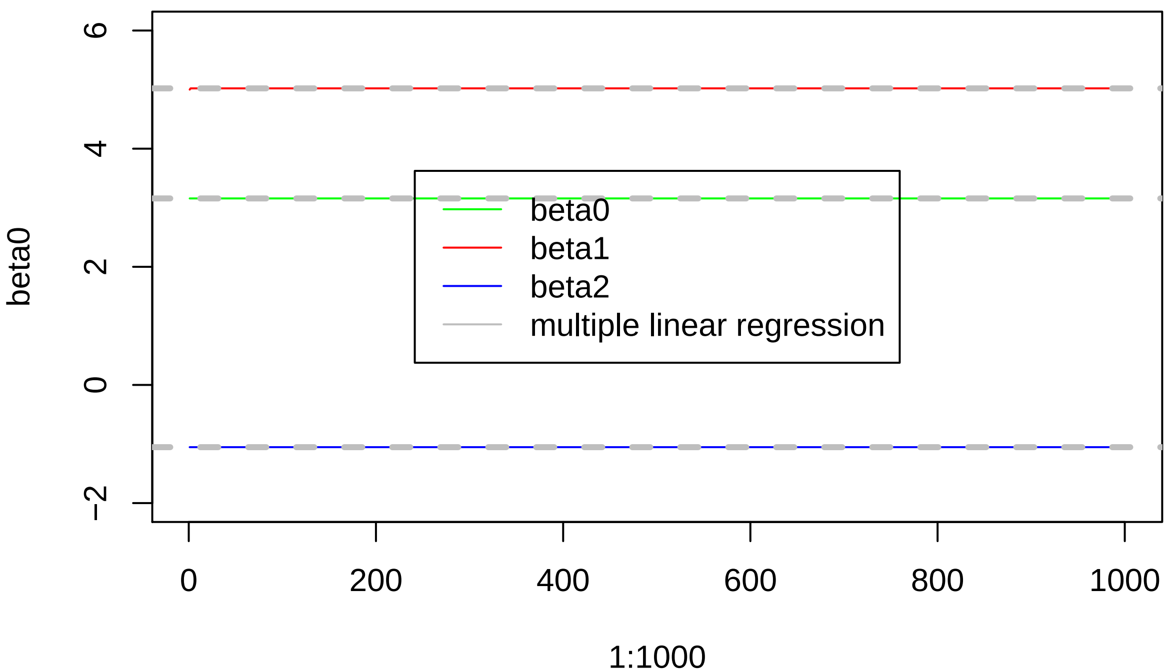
```
lm.fit = lm(Y~X1+X2)
lm.fit$coefficients
```

```
## (Intercept)          X1          X2
##    3.157710    5.021110   -1.053467
```

```r
plot(1:1000, beta0, type = "line", col="green",  ylim = c(-2, 6))
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to
## first character
```

```r
lines(1:1001, beta1, col="red")
lines(1:1000, beta2, col="blue")


abline(h = lm.fit$coefficients[1], lty = "dashed", lwd = 3, col = "grey")
abline(h = lm.fit$coefficients[2], lty = "dashed", lwd = 3, col = "grey")
abline(h = lm.fit$coefficients[3], lty = "dashed", lwd = 3, col = "grey")


legend("center", c("beta0", "beta1", "beta2", "multiple linear regression"), lty = 1, col = c("green",
```



(g)

After one backfitting iteration coefficients have not changed