**Chapter 7: Moving Beyond Linearity**

**1(a)**

时，



。

**1(b)**

时，



。

**1(c)**

时，



即

时，



所以，即在点连续。

**1(d)**

时，



时，



时，



时，



所以，即在点连续。

**1(e)**

时，



时，



时，



时，



所以，即在点连续。

**2(a)**

****时，****时整体能取到最小值，从而****。

**2(b)**

****时，同理可得，其中，为任意实数。又要使取得最小值，从而。

即

**2(c)**

****时，同理可得，其中，，为任意实数。又要使取得最小值，从而，。

**2(d)**

****时，同理可得，其中，，，为任意实数。又要使取得最小值，从而，。







**2(e)**

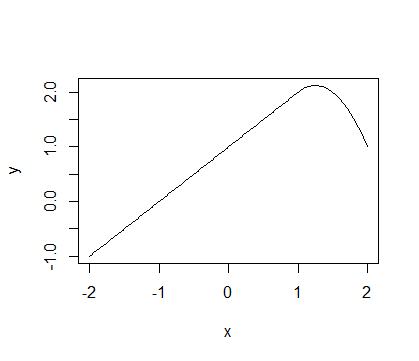
****时，同**2(d)。**

**3**

> x = seq(-2,2,0.02)

> y = 1+x-2\*(x-1)^2\*I(x>1)

> plot(x,y,type = "l")



**4**

时，

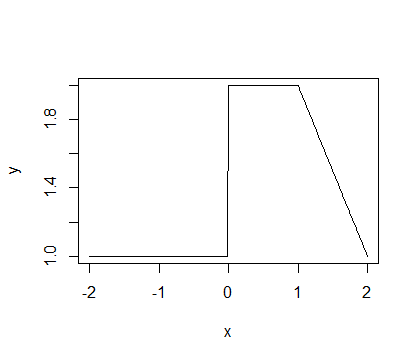
时，

时，

> x = seq(-2,2,0.02)

> y=1+1\*(I(x>=0)\*I(x<=2)-(x-1)\*(x>=1)\*I(x<=2))+3\*((x-3)\*(x>=3)\*I(x<=4)+(x>4)\*I(x<=5))

> plot(x,y,type = "l")



**5(a)**

有更小的training RSS，因为是包含更高次项的函数，更加flexible，所以training RSS更小。

**5(b)**

有更小的test RSS吧，可能会出现过拟合的问题。

**5(c)**

此时和是一样的。

**6(a)**

> library(boot)

> set.seed(1)

> cv.errors=rep(NA,10)

> for (i in 1:10) {

+ glm.fit1=glm(wage~poly(age,i),data = Wage)

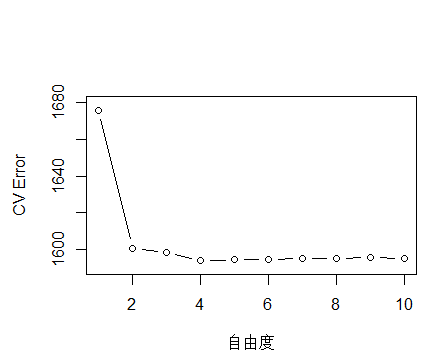
+ cv.errors[i]=cv.glm(Wage,glm.fit1,K=10)$delta[2]

+ }

> plot(c(1:10),cv.errors,type = "b",xlab = "自由度",ylab = "CV Error",ylim = c(1590,1680))

> which.min(cv.errors)

[1] 4



四次多项式模型交叉验证误差最小。

> fit.1 = lm(wage~poly(age, 1), data=Wage)

> fit.2 = lm(wage~poly(age, 2), data=Wage)

> fit.3 = lm(wage~poly(age, 3), data=Wage)

> fit.4 = lm(wage~poly(age, 4), data=Wage)

> fit.5 = lm(wage~poly(age, 5), data=Wage)

> fit.6 = lm(wage~poly(age, 6), data=Wage)

> fit.7 = lm(wage~poly(age, 7), data=Wage)

> fit.8 = lm(wage~poly(age, 8), data=Wage)

> fit.9 = lm(wage~poly(age, 9), data=Wage)

> fit.10 = lm(wage~poly(age, 10), data=Wage)

> anova(fit.1, fit.2, fit.3, fit.4, fit.5, fit.6, fit.7, fit.8, fit.9, fit.10)

Analysis of Variance Table

Model 1: wage ~ poly(age, 1)

Model 2: wage ~ poly(age, 2)

Model 3: wage ~ poly(age, 3)

Model 4: wage ~ poly(age, 4)

Model 5: wage ~ poly(age, 5)

Model 6: wage ~ poly(age, 6)

Model 7: wage ~ poly(age, 7)

Model 8: wage ~ poly(age, 8)

Model 9: wage ~ poly(age, 9)

Model 10: wage ~ poly(age, 10)

Res.Df RSS Df Sum of Sq F Pr(>F)

1 2998 5022216

2 2997 4793430 1 228786 143.7638 < 2.2e-16 \*\*\*

3 2996 4777674 1 15756 9.9005 0.001669 \*\*

4 2995 4771604 1 6070 3.8143 0.050909 .

5 2994 4770322 1 1283 0.8059 0.369398

6 2993 4766389 1 3932 2.4709 0.116074

7 2992 4763834 1 2555 1.6057 0.205199

8 2991 4763707 1 127 0.0796 0.777865

9 2990 4756703 1 7004 4.4014 0.035994 \*

10 2989 4756701 1 3 0.0017 0.967529

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

三次或四次的多项式模型可以更好地拟合数据。

>agelims=range(age)

>age.grid=seq(from=agelims[1],to=agelims[2])

>preds=predict(fit.4,newdata =list(age=age.grid),se=TRUE)

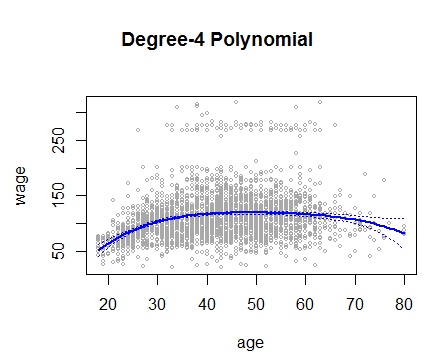
>se.bands=cbind(preds$fit+2\*preds$se.fit,preds$fit-2\*preds$se.fit)

> plot(Wage$age,Wage$wage,xlim=agelims,cex =.5,col ="darkgrey",xlab = "age",ylab = "wage")

> title("Degree-4 Polynomial",outer=T)

> lines(age.grid,preds$fit,lwd =2,col ="blue")

> matlines(age.grid,se.bands,lwd =1,col ="blue",lty =3)



**6(b)**

> step.errors=rep(NA,9)

> for(i in 2:10) {

+ Wage$age.cut = cut(Wage$age,i)

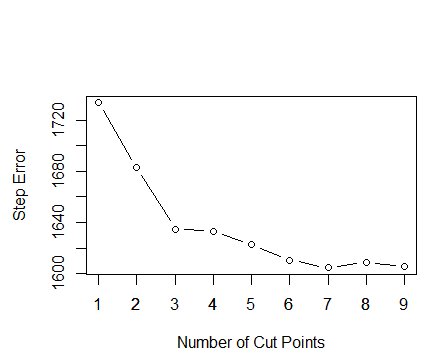
+ glm.fit2= glm(wage~age.cut,data=Wage)

+ step.errors[i-1] = cv.glm(Wage,glm.fit2, K=10)$delta[2]

+ }

> plot(c(2:10),step.errors,type = "b",xlab = "自由度",ylab = "Step Error")

axis(side = 1,c(1:9),labels = c(1:9))



> which.min(step.errors)

[1] 7

七个分割点时误差最小。

fit.8 = glm(wage~cut(age,8),data=Wage)

> agelims=range(age)

> age.grid=seq(from=agelims[1],to=agelims[2])

> preds=predict(fit.8,newdata =list(age=age.grid),se=TRUE)

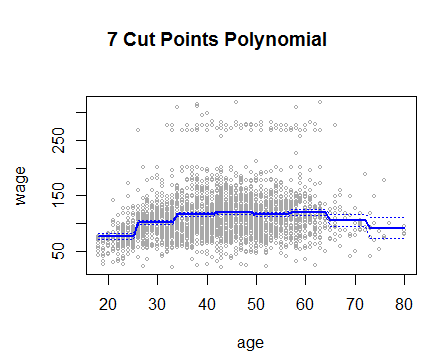
> se.bands=cbind(preds$fit+2\*preds$se.fit,preds$fit-2\*preds$se.fit)

> plot(Wage$age,Wage$wage,xlim=agelims,cex =.5,col ="darkgrey",xlab = "age",ylab = "wage")

> title("7 Cut Points Polynomial",outer=T)

> lines(age.grid,preds$fit,lwd =2,col ="blue")

> matlines(age.grid,se.bands,lwd =1,col ="blue",lty =3)

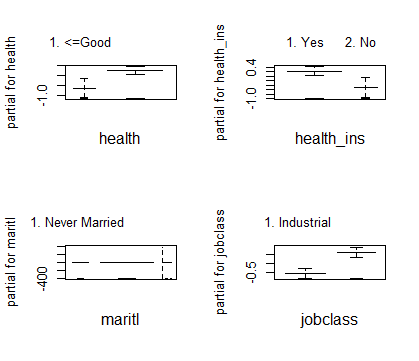


**7**

> gam.fit7=gam(I(wage >250)~health+health\_ins+maritl+jobclass,family=binomial,data=Wage)

> par(mfrow =c(2,2))

> plot(gam.fit7,se=T,col =" green ")



> table(health,I(wage>250))

health FALSE TRUE

1. <=Good 848 10

2. >=Very Good 2073 69

> table(health\_ins,I(wage>250))

health\_ins FALSE TRUE

1. Yes 2015 68

2. No 906 11

> table(maritl,I(wage>250))

maritl FALSE TRUE

1. Never Married 643 5

2. Married 2003 71

3. Widowed 19 0

4. Divorced 202 2

5. Separated 54 1

> table(jobclass,I(wage>250))

jobclass FALSE TRUE

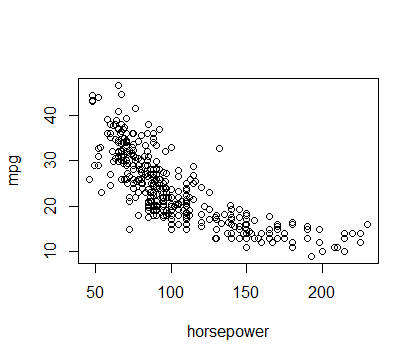
1. Industrial 1526 18

2. Information 1395 61

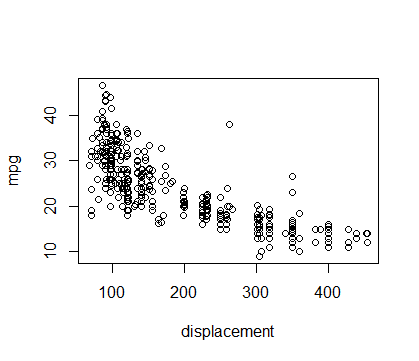
健康状况更好的人，有健康保险的人，已婚的人及从事信息行业工作的人有更多和更高比例的人获得更高的工资。

**8**

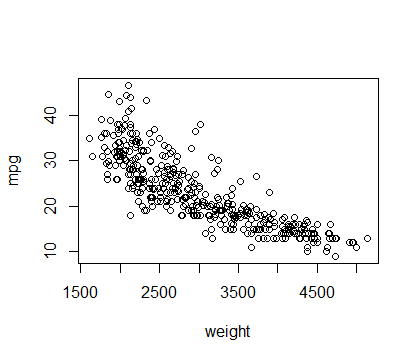
> plot(horsepower,mpg)



> plot(displacement,mpg)



> plot(weight,mpg)



> lm.fit8 = list()

> for (d in 1:10) {

+ lm.fit8[[d]] = lm(mpg ~ poly(displacement, d), data = Auto)

+ }

> lm.fit8 = list()

> for (d in 1:5) {

+ lm.fit8[[d]] = lm(mpg ~ poly(displacement, d), data = Auto)

+ }

> anova(lm.fit8[[1]],lm.fit8[[2]],lm.fit8[[3]],lm.fit8[[4]],lm.fit8[[5]])

Analysis of Variance Table

Model 1: mpg ~ poly(displacement, d)

Model 2: mpg ~ poly(displacement, d)

Model 3: mpg ~ poly(displacement, d)

Model 4: mpg ~ poly(displacement, d)

Model 5: mpg ~ poly(displacement, d)

Res.Df RSS Df Sum of Sq F Pr(>F)

1 390 8378.8

2 389 7412.3 1 966.56 50.5487 5.668e-12 \*\*\*

3 388 7392.3 1 19.94 1.0429 0.3078

4 387 7391.7 1 0.60 0.0314 0.8595

5 386 7380.8 1 10.88 0.5692 0.4510

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> lm.fit8\_ = list()

> for (d in 1:5) {

+ lm.fit8\_[[d]] = lm(mpg ~ poly(horsepower, d), data = Auto)

+ }

> anova(lm.fit8\_[[1]],lm.fit8\_[[2]],lm.fit8\_[[3]],lm.fit8\_[[4]],lm.fit8\_[[5]])

Analysis of Variance Table

Model 1: mpg ~ poly(horsepower, d)

Model 2: mpg ~ poly(horsepower, d)

Model 3: mpg ~ poly(horsepower, d)

Model 4: mpg ~ poly(horsepower, d)

Model 5: mpg ~ poly(horsepower, d)

Res.Df RSS Df Sum of Sq F Pr(>F)

1 390 9385.9

2 389 7442.0 1 1943.89 103.8767 < 2.2e-16 \*\*\*

3 388 7426.4 1 15.59 0.8333 0.361897

4 387 7399.5 1 26.91 1.4382 0.231169

5 386 7223.4 1 176.15 9.4131 0.002306 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> lm.fit8\_\_ = list()

> for (d in 1:5) {

+ lm.fit8\_\_[[d]] = lm(mpg ~ poly(weight, d), data = Auto)

+ }

> anova(lm.fit8\_\_[[1]],lm.fit8\_\_[[2]],lm.fit8\_\_[[3]],lm.fit8\_\_[[4]],lm.fit8\_\_[[5]])

Analysis of Variance Table

Model 1: mpg ~ poly(weight, d)

Model 2: mpg ~ poly(weight, d)

Model 3: mpg ~ poly(weight, d)

Model 4: mpg ~ poly(weight, d)

Model 5: mpg ~ poly(weight, d)

Res.Df RSS Df Sum of Sq F Pr(>F)

1 390 7321.2

2 389 6784.9 1 536.34 30.6147 5.817e-08 \*\*\*

3 388 6784.8 1 0.05 0.0028 0.9580

4 387 6777.0 1 7.88 0.4500 0.5027

5 386 6762.3 1 14.67 0.8374 0.3607

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> lm.fit=lm(mpg ~ poly(displacement,2)+poly(weight,2)+poly(horsepower,2),data = Auto)

> summary(lm.fit)

Call:

lm(formula = mpg ~ poly(displacement, 2) + poly(weight, 2) +

poly(horsepower, 2), data = Auto)

Residuals:

Min 1Q Median 3Q Max

-11.7507 -2.1972 -0.2779 2.0043 15.4606

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 23.4459 0.1973 118.822 < 2e-16 \*\*\*

poly(displacement, 2)1 -20.1433 12.7914 -1.575 0.11613

poly(displacement, 2)2 17.7385 7.5018 2.365 0.01855 \*

poly(weight, 2)1 -53.4821 12.6177 -4.239 2.82e-05 \*\*\*

poly(weight, 2)2 8.1038 5.8596 1.383 0.16747

poly(horsepower, 2)1 -59.3958 10.2600 -5.789 1.47e-08 \*\*\*

poly(horsepower, 2)2 15.7790 5.7277 2.755 0.00615 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.907 on 385 degrees of freedom

Multiple R-squared: 0.7533, Adjusted R-squared: 0.7495

F-statistic: 195.9 on 6 and 385 DF, p-value: < 2.2e-16

> lm.fit=lm(mpg ~ I(displacement^2)+weight+poly(horsepower,2),data = Auto)

> summary(lm.fit)

Call:

lm(formula = mpg ~ I(displacement^2) + weight + poly(horsepower,

2), data = Auto)

Residuals:

Min 1Q Median 3Q Max

-10.6613 -2.2913 -0.2182 2.0325 14.9584

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.657e+01 1.642e+00 22.274 < 2e-16 \*\*\*

I(displacement^2) -4.875e-06 1.297e-05 -0.376 0.707

weight -4.328e-03 6.551e-04 -6.607 1.30e-10 \*\*\*

poly(horsepower, 2)1 -5.296e+01 1.006e+01 -5.266 2.32e-07 \*\*\*

poly(horsepower, 2)2 3.103e+01 4.776e+00 6.495 2.55e-10 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.002 on 387 degrees of freedom

Multiple R-squared: 0.7397, Adjusted R-squared: 0.737

F-statistic: 275 on 4 and 387 DF, p-value: < 2.2e-16

> lm.fit=lm(mpg ~ weight+poly(horsepower,2),data = Auto)

> summary(lm.fit)

Call:

lm(formula = mpg ~ weight + poly(horsepower, 2), data = Auto)

Residuals:

Min 1Q Median 3Q Max

-10.7687 -2.3002 -0.1931 2.0091 14.9555

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.680e+01 1.529e+00 24.069 < 2e-16 \*\*\*

weight -4.483e-03 5.089e-04 -8.809 < 2e-16 \*\*\*

poly(horsepower, 2)1 -5.504e+01 8.402e+00 -6.551 1.82e-10 \*\*\*

poly(horsepower, 2)2 3.024e+01 4.296e+00 7.040 8.77e-12 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.998 on 388 degrees of freedom

Multiple R-squared: 0.7396, Adjusted R-squared: 0.7376

F-statistic: 367.4 on 3 and 388 DF, p-value: < 2.2e-16

分别单独用displacement，weight和horsepower作为预测变量进行回归分析时，发现每个预测变量的一次和二次项系数统计上都是显著不为0的，但是将三个预测变量放到同一个模型里一起进行考虑时，displacement的一次项和二次项系数统计上都不是显著不为0的了，weight只有一次向系数统计上显著不为0，horsepower一次项和二次项系数仍然显著不为0，即horsepower与mpg存在非线性关系。

> cut.fit=lm(mpg~cut(weight,3),data = Auto)

> summary(cut.fit)

Call:

lm(formula = mpg ~ cut(weight, 3), data = Auto)

Residuals:

Min 1Q Median 3Q Max

-11.3604 -3.3604 -0.4422 2.3578 18.3578

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 29.3604 0.3509 83.67 <2e-16 \*\*\*

cut(weight, 3)(2.79e+03,3.96e+03] -9.7182 0.5461 -17.80 <2e-16 \*\*\*

cut(weight, 3)(3.96e+03,5.14e+03] -15.4850 0.6977 -22.19 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.862 on 389 degrees of freedom

Multiple R-squared: 0.6139, Adjusted R-squared: 0.6119

F-statistic: 309.3 on 2 and 389 DF, p-value: < 2.2e-16

将weight切分成三段，每段系数统计上都是显著不为0的。

**9(a)**

> lm.fit9=lm(nox~poly(dis,3),data=Boston)

> summary(lm.fit9)

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

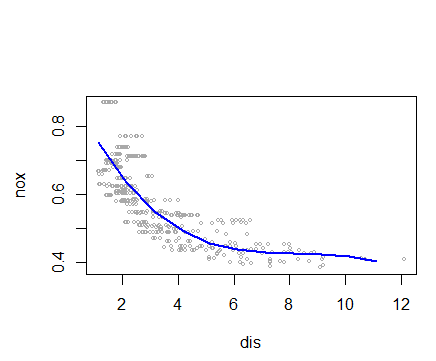
> dislims=range(dis)

> dis.grid=seq(from=dislims[1],to=dislims[2])

> preds=predict(lm.fit9,newdata =list(dis=dis.grid),se=TRUE)

> plot(Boston$dis,Boston$nox,xlim=dislims,cex =.5,col ="darkgrey",xlab = "dis",ylab = "nox")

> lines(dis.grid,preds$fit,lwd =2,col ="blue")



**9(b)**

> RSS = rep(NA, 10)

> for (i in 1:10) {

+ lm.fit9= lm(nox ~ poly(dis, i), data = Boston)

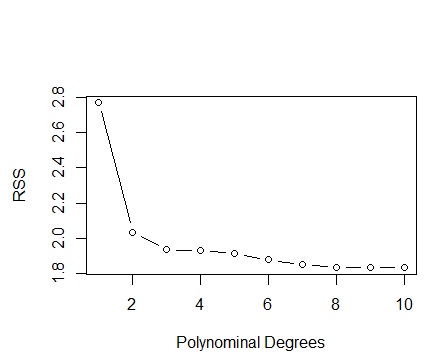
+ RSS[i] = sum(lm.fit9$residuals^2)

+ }

> RSS

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

plot(c(1:10),RSS,xlab = "Polynominal Degrees",ylab = "RSS",type = "b")



**9(c)**

> library(boot)

> set.seed(1)

> cv.errors=rep(NA,10)

> for (i in 1:10) {

+ glm.fit9=glm(nox~poly(dis,i),data = Boston)

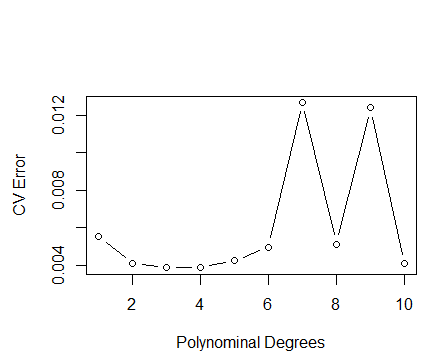
+ cv.errors[i]=cv.glm(Boston,glm.fit9,K=10)$delta[2]

+ }

> plot(c(1:10),cv.errors,type = "b",xlab = "Polynominal Degrees",ylab = "CV Error")

> which.min(cv.errors)

[1] 4



**9(d)**

> library(splines)

> bs.fit9=lm(nox~bs(dis,df=4),data = Boston)

> summary(bs.fit9)

Call:

lm(formula = nox ~ bs(dis, df = 4), data = Boston)

Residuals:

Min 1Q Median 3Q Max

-0.124622 -0.039259 -0.008514 0.020850 0.193891

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 0.73447 0.01460 50.306 < 2e-16 \*\*\*

bs(dis, df = 4)1 -0.05810 0.02186 -2.658 0.00812 \*\*

bs(dis, df = 4)2 -0.46356 0.02366 -19.596 < 2e-16 \*\*\*

bs(dis, df = 4)3 -0.19979 0.04311 -4.634 4.58e-06 \*\*\*

bs(dis, df = 4)4 -0.38881 0.04551 -8.544 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06195 on 501 degrees of freedom

Multiple R-squared: 0.7164, Adjusted R-squared: 0.7142

F-statistic: 316.5 on 4 and 501 DF, p-value: < 2.2e-16

> attr(bs(age,df=4),"knots")

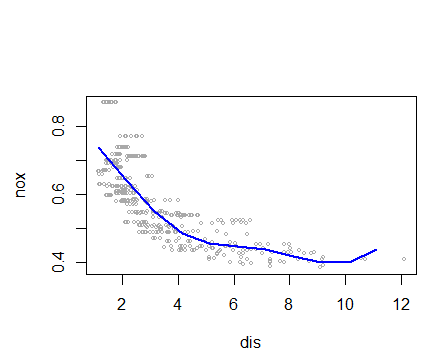
50%

49

> bs.pred=predict(bs.fit9,newdata =list(dis=dis.grid),se=TRUE)

> plot(Boston$dis,Boston$nox,xlim=dislims,cex =.5,col ="darkgrey",xlab = "dis",ylab = "nox")

> lines(dis.grid,bs.pred$fit,lwd =2,col ="blue")



**9(e)**

> spline.RSS = rep(NA, 8)

> for (i in 3:10) {

+ bs.fit9 = lm(nox ~ bs(dis, df = i), data = Boston)

+ spline.RSS[i-2] = sum(bs.fit9$residuals^2)

+ }

> spline.RSS

[1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535

> which.min(spline.RSS)

[1] 8

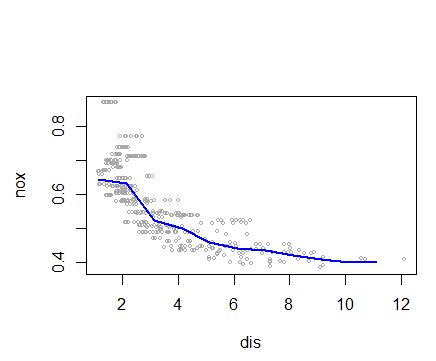
自由度为10时RSS最小。

> bs.fit9 = lm(nox ~ bs(dis, df = 10), data = Boston)

> bs.pred=predict(bs.fit9,newdata =list(dis=dis.grid),se=TRUE)

> plot(Boston$dis,Boston$nox,xlim=dislims,cex =.5,col ="darkgrey",xlab = "dis",ylab = "nox")

> lines(dis.grid,bs.pred$fit,lwd =2,col ="blue")



> spline.RSS1 = rep(NA, 10)

> for (i in 1:10) {

+ ns.fit9 = lm(nox ~ ns(dis, df = i), data = Boston)

+ spline.RSS1[i] = sum(ns.fit9$residuals^2)

+ }

> spline.RSS1

[1] 2.768563 1.974579 1.930501 1.885805 1.860232 1.854157 1.848602 1.797749 1.798482 1.789243

> which.min(spline.RSS1)

[1] 10

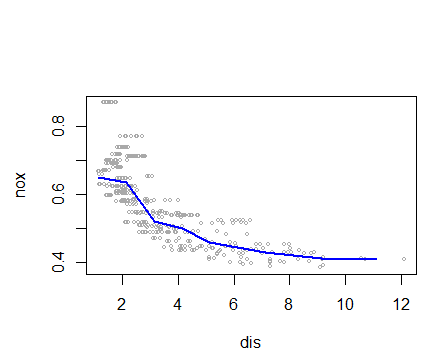
自由度为10时RSS最小。

> ns.fit9 = lm(nox ~ ns(dis, df = 10), data = Boston)

> ns.pred=predict(ns.fit9,newdata =list(dis=dis.grid),se=TRUE)

> plot(Boston$dis,Boston$nox,xlim=dislims,cex =.5,col ="darkgrey",xlab = "dis",ylab = "nox")

> lines(dis.grid,ns.pred$fit,lwd =2,col ="blue")



**9(f)**

> set.seed(1)

> spline.cv.errors = rep(NA, 10)

> for (i in 1:10) {

+ lm.fit = glm(nox ~ ns(dis, df = i), data = Boston)

+ spline.cv.errors[i] = cv.glm(Boston, lm.fit, K = 10)$delta[2]

+ }

> which.min(spline.cv.errors)

[1] 10

自由度为10时交叉验证误差最小。

**10(a)**

> set.seed(1)

> library(leaps)

> attach(College)

> train = sample(1:nrow(College), nrow(College)/2)

> test = -train

> College.train = College[train, ]

> College.test = College[test, ]

> reg.fit10=regsubsets(Outstate ~ ., data = College.train, nvmax = 17, method = "forward")

> reg.summary10=summary(reg.fit10)

> which.min(reg.summary10$cp)

[1] 13

> which.min(reg.summary10$bic)

[1] 6

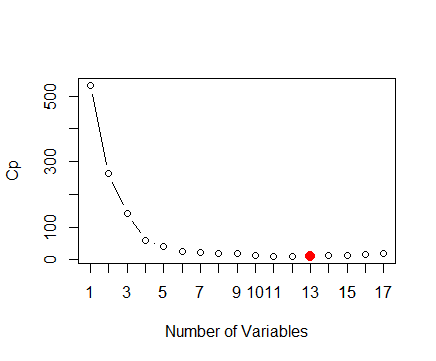
> which.max(reg.summary10$adjr2)

[1] 13

> plot(reg.summary10$cp ,xlab =" Number of Variables ",ylab="Cp",type="b")

> axis(side = 1,at=c(1:17),labels = c(1:17))

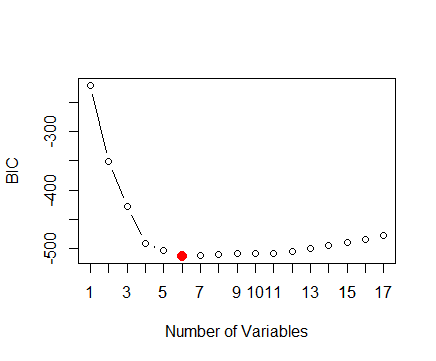
> points (13, reg.summary10$cp [13], col ="red",cex =2, pch =20)



> plot(reg.summary10$bic,xlab="Number of Variables",ylab="BIC",type="b")

> axis(side = 1,at=c(1:17),labels = c(1:17))

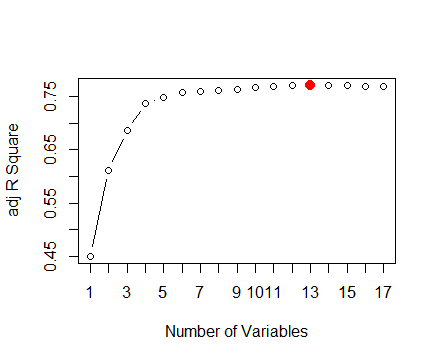
> points (6,reg.summary10$bic[6], col ="red",cex =2, pch =20)



>plot(reg.summary10$adjr2,xlab="Number of Variables",ylab="adj R Square",type="b")

> axis(side = 1,at=c(1:17),labels = c(1:17))

> points (13,reg.summary10$adjr2[13], col ="red",cex =2, pch =20)



选择6个预测变量比较合适。

> coef(reg.fit10,6)

(Intercept) PrivateYes Room.Board Terminal perc.alumni Expend Grad.Rate

-4241.4402916 2790.4303173 0.9629335 37.8412517 60.6406044 0.2149396 30.3831268

**10(b)**

**11(a) (b)**

> set.seed(1)

> X1 = rnorm(100,2,1)

> X2 = rnorm(100,1,2)

> eps = rnorm(100)

> Y = 1 + 2\*X1 + 3\*X2 + eps

**11(c)**

> beta1=2

> a=Y-beta1\*X1

> beta2=lm(a~X2)$coef[2]

> beta2

X2

2.973257

**11(d)**

> a=Y-beta2\*X2

> beta1=lm(a~X1)$coef[2]

> beta1

X1

2.02111

**11(e)**

> beta0 = rep(NA, 1000)

> beta1 = rep(NA, 1000)

> beta2 = rep(NA, 1000)

> beta1[1] = 5

> for (i in 1:1000){

+ a = Y-beta1[i]\*X1

+ beta2[i]=lm(a~X2)$coef[2]

+ a=Y-beta2[i]\*X2

+ fit=lm(a~X1)

+ if (i < 1000){

+ beta1[i+1]=fit$coef[2]

+ }

+ beta0[i]=fit$coef[1]

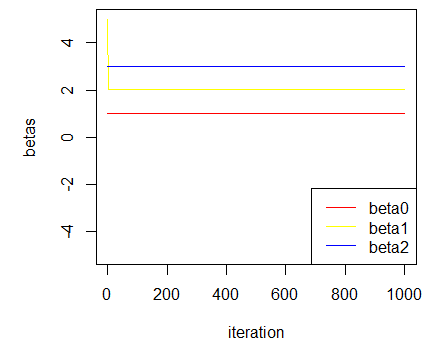
+ }

> plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-5,5), col = "red")

> lines(1:1000, beta1, col = "yellow")

> lines(1:1000, beta2, col = "blue")

> legend("bottomright", c("beta0", "beta1", "beta2"), lty = 1, col = c("red","yellow","blue"))



**11(f)**

> fit1 = lm(Y ~ X1 + X2)

> plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-5,5), col = "red")

> lines(1:1000, beta1, col = "yellow")

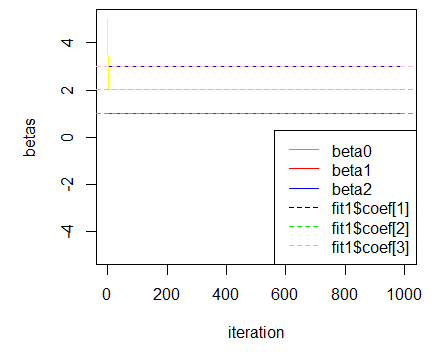
> lines(1:1000, beta2, col = "blue")

> abline(h = fit1$coef[1], lty = 2, col = "green")

> abline(h = fit1$coef[2], lty = 2, col = "gray")

> abline(h = fit1$coef[3], lty = 2, col = "pink")

> legend("bottomright", c("beta0", "beta1", "beta2", "fit1$coef[1]","fit1$coef[2]","fit1$coef[3]"), lty = c(1,1,1,2,2,2), col = c("green", "red", "blue", "black","green","gray","pink"))



**11(g)**

一次就够了。

**12**

> set.seed(1)

> p = 100

> n = 1000

> eps= rnorm(n)

> x = matrix(ncol = p, nrow = n)

> coeffi = rep(NA, p)

> for (i in 1:p) {

+ x[, i] = rnorm(n)

+ coeffi[i] = rnorm(1) \* 100

+ }

> y = x %\*% coeffi + eps

> fit12=lm(y~x)

> beta = rep(0, p)

> maxiter = 1000

> errors = rep(NA, maxiter + 1)

> iter = 2

> errors[1] = Inf

> errors[2] = sum((y - x %\*% beta)^2)

> threshold = 1e-05

> while (iter <= maxiter && errors[iter-1]-errors[iter] > threshold) {

+ for (i in 1:p) {

+ a = y - x[,-i]%\*%beta[-i]

+ beta[i] = lm(a ~ x[,i])$coef[2]

+ }

+ iter = iter + 1

+ errors[iter] = sum((fit12$coefficients[-1]-beta)^2)

+ }

> print(iter-2)

[1] 10

迭代了10次。

> plot(c(1:10),errors[3:12],xlab = "iterations",ylab = "errors X10^-5")

> axis(side = 1,at=c(1:10),labels = c(1:10))

