高级统计方法 第6次作业:

序号： 08 姓名：潘晨楷 学号：20212241116 班级：软2107

**第五章**

**概念**

2.问题（略）

（a）问题（略）

1-1/n

（b）问题（略）

1-1/n

（c）问题（略）

（1-1/n）^n

自助法是有放回的，且每次抽样都是独立概率。

（d）问题（略）

1-（1-1/5）^5 = 67.2%

（e）问题（略）

1−(1−1/100)^100=1−(99/100)^100=63.4%

（f）问题（略）

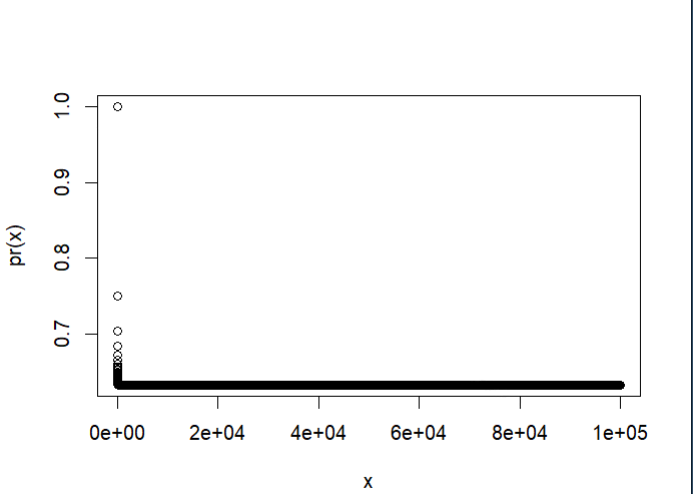
1−(1−1/10000)^10000=1−(9999/10000)^10000=63.4%

（g）问题（略）

pr = function(n) return(1 - (1 - 1/n)^n)

x = 1:1e+05

plot(x, pr(x))



随着x的变小，pr(x)基本稳定在了63.2%的概率上。

（h）问题（略）

store = rep(NA, 10000)

for(i in 1:10000){}

for(i in 1:10000){ store[i] = sum(sample(1:100, rep = TRUE) == 4) > 0}



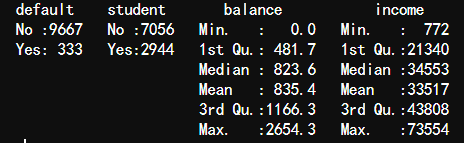
发现第四个观测是否包含在自助法样本里的概率为0.6412.

**应用**

1. 问题（略）

library(ISLR)

summary(Default)



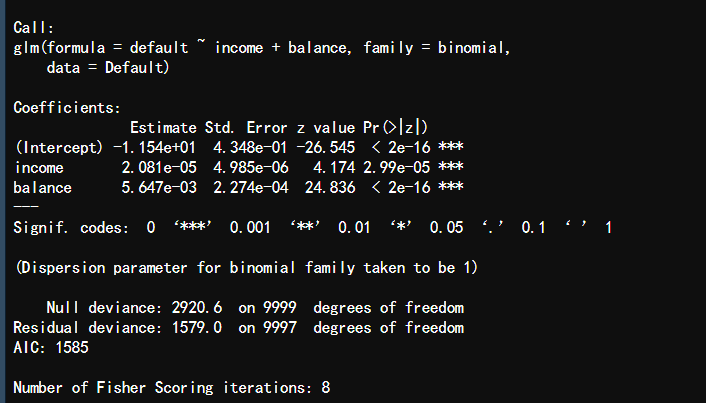
attach(Default)

（a）问题（略）

set.seed(1)

glm.fit = glm(default ~ income + balance, data = Default, family = binomial)

summary(glm.fit)



（b）问题（略）

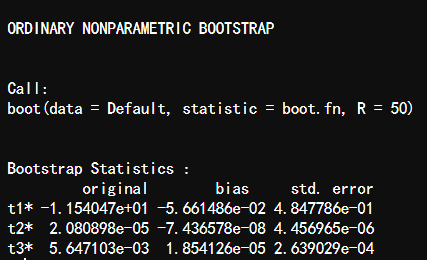
boot.fn = function(data, index) return(coef(glm(default ~ income + balance,

data = data, family = binomial, subset = index)))

（c）问题（略）

library(boot)

boot(Default, boot.fn, 50)



（d）问题（略）

根据结果对比可得，SE(),SE()的自助法和glm()函数所得标准误差比较 接近。

8.问题（略）

（a）问题（略）

set.seed(1)

y = rnorm(100)

x = rnorm(100)

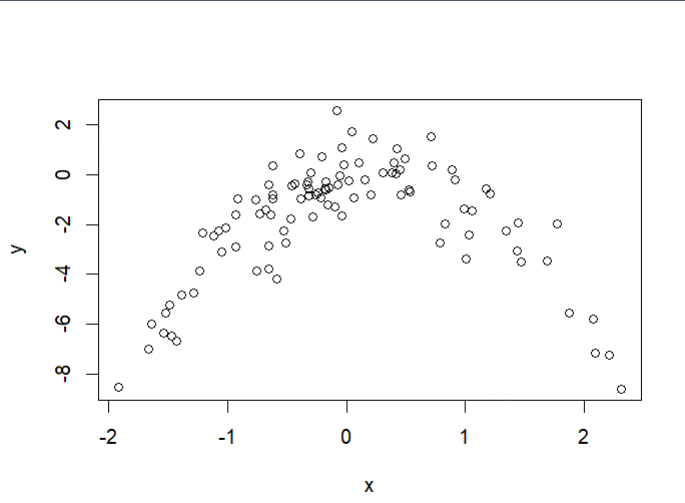
y = x - 2 \* x^2 + rnorm(100)

n = 100, p = 2.

Y=X−2X^2+ϵ.

（b）问题（略）

plot(x, y)



x的大致取值范围是-2到2，y的大致取值范围是-8到2，整体呈抛物线分布。

（c）问题（略）

library(boot)

Data = data.frame(x, y)

set.seed(1)

# i.

glm.fit = glm(y ~ x)

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 2))

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 3))

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 4))

cv.glm(Data, glm.fit)$delta



（d）问题（略）

set.seed(10)

glm.fit = glm(y ~ x)

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 2))

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 3))

cv.glm(Data, glm.fit)$delta



glm.fit = glm(y ~ poly(x, 4))

cv.glm(Data, glm.fit)$delta



一样。因为LOOCV方法在训练集和验证集的分割上不存在随机性。在c和d中都是对同一个模型的同一个观测。

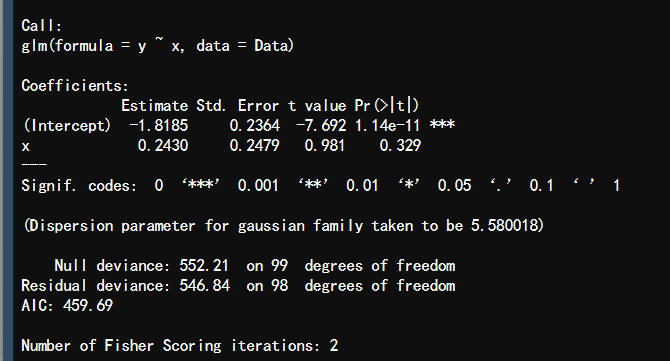
（e）问题（略）

二次模型具有最小的LOOCV误差，因为实际模型也是二次的。

（f）问题（略）

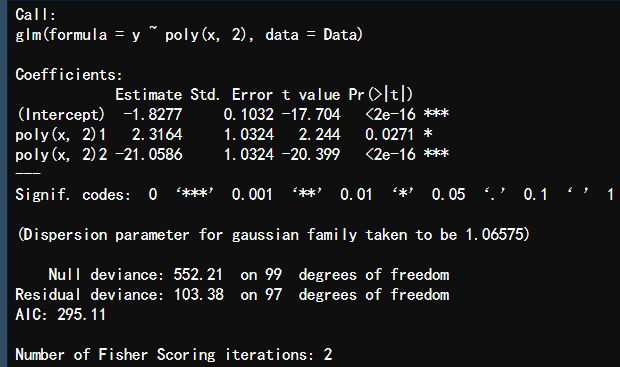
glm.fit=glm(y~x,data=Data)

summary(glm.fit)



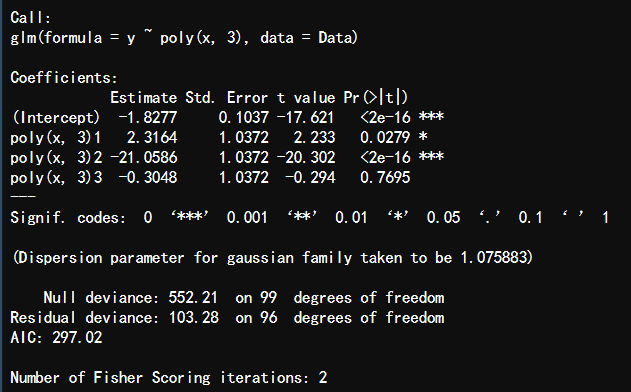
glm.fit=glm(y~ poly(x,2) ,data=Data)

summary(glm.fit)



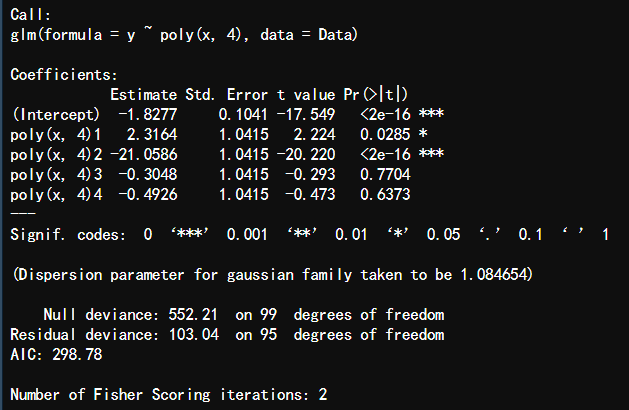
glm.fit=glm(y~ x+x2+x3 ,data=Data)

summary(glm.fit)



glm.fit=glm(y~ poly(x,4) ,data=Data)

summary(glm.fit)

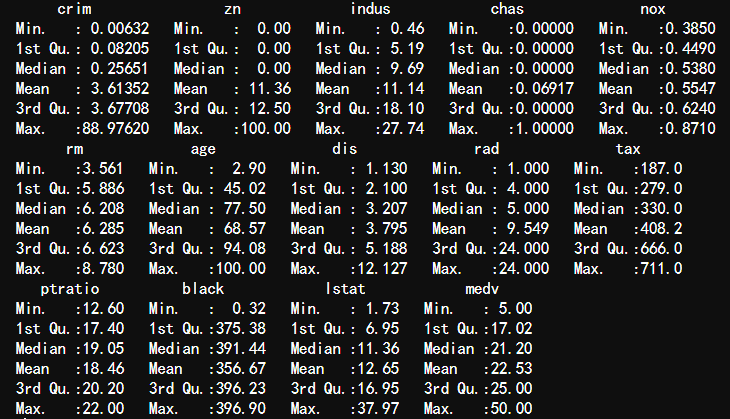


常数项，一次项和二次项具有统计上的显著性，与交叉验证法所得到的结论一致。

9.问题（略）

library(MASS)

summary(Boston)



set.seed(1)

attach(Boston)

1. 问题（略）

medv.mean = mean(medv)

medv.mean



1. 问题（略）

medv.err = sd(medv)/sqrt(length(medv))

medv.err



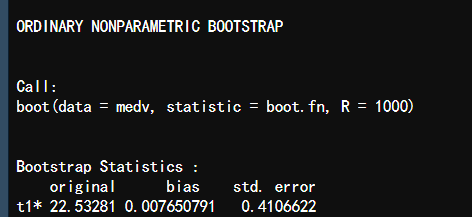
1. 问题（略）

boot.fn = function(data, index) return(mean(data[index]))

library(boot)

bstrap = boot(medv, boot.fn, 1000)

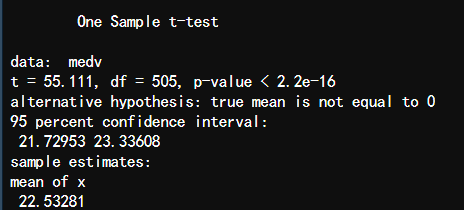
Bstrap



Bootstrap得到的标准误差为0.4106与b中得到的0.4088较为接近。

1. 问题（略）

t.test(medv)



c(bstrap$t0 - 2 \* 0.4119, bstrap$t0 + 2 \* 0.4119)



得到的两个置信区间结果非常相似。

[21.72,23.33]和[21.70,23.35]

1. 问题（略）

medv.med = median(medv)

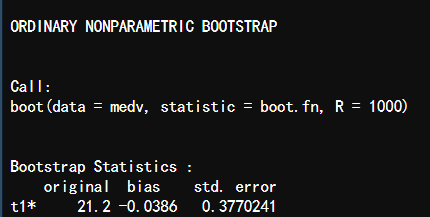
medv.med



1. 问题（略）

boot.fn = function(data, index) return(median(data[index]))

boot(medv, boot.fn, 1000)



中位数一致，为21.2，标准误差为0.377，较小。

1. 问题（略）

medv.tenth = quantile(medv, c(0.1))

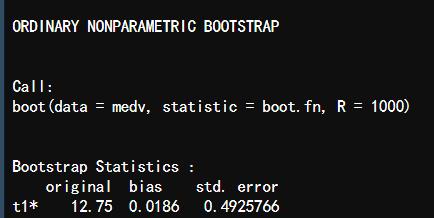
medv.tenth



1. 问题（略）

boot.fn = function(data, index) return(quantile(data[index], c(0.1)))

boot(medv, boot.fn, 1000)



标准误差为0.492, 可以认为误差较小。

**第六章**

**概念**

1.问题（略）

（a）问题（略）

最佳子集选择具有最小的训练RSS。因为其他两种方法确定具有路径依赖的模型时，它们首先选择第一个迭代到第k个模型的预测器。

（b）问题（略）

最佳子集选择可能具有最小的测试RSS，因为它比其他方法考虑更多的模型。

（c）问题（略）

i.对 ii. 对 iii. 错 iv. 错 v. 错

2.问题（略）

（a）问题（略）

iii是正确的，因为更少的方差，更多的偏差能创造更少的灵活性和更好的预测。

（b）问题（略）

iii是正确的，因为更少的方差，更多的偏差能创造更少的灵活性和更好的预测。

（c）问题（略）

ii是正确的，更加灵活会导致更少的偏差和更大的差异。

3.问题（略）

（a）问题（略）

iv.稳定减小。当s从0开始增加，β也从0开始逐渐增加到接近最小二乘法的系数估计值，所以训练集RSS从开始时的最大逐渐减小到最小二乘法得出的训练集RSS。

（b）问题（略）

ii.最初减小，然后开始增加，图像呈现一个U形。一开始s为0的时候，模型是欠拟合的，具有很高的测试集RSS，当s开始增加，模型会由于拟合效果越来越好，测试集RSS会减小，但当s过大的时候，会出现过拟合现象，测试集RSS又会增加。

（c）问题（略）

iii.稳定增长。在s为0的时候，模型估计的都是常数，方差几乎为0，当s开始增加，方差也会增加。

（d）问题（略）

iv.稳定减小。在s为0的时候，模型估计的都是常数，与实际值相比有着很大的偏差，而随着s的增加，拟合效果越来越好，偏差便越来越小。

（e）问题（略）

v.保持不变。根据不可约误差的定义，其与模型无关，因此s的值的变化不会引起不可约误差的变化，所以其保持不变。

4.问题（略）

（a）问题（略）

iii.稳定增加。当λ从0开始增加，其系数估计值从最小二乘法的估计值开始逐步减少到0，所以其训练集RSS从最小值逐步增加。

（b）问题（略）

ii.最初减小，然后开始增加，图像呈现一个U形。λ为0时，系数是最小二乘法的估计值，该模型有过拟合现象，有着较大的测试集RSS，而随着λ的增加，过拟合现象得到缓解，测试集RSS减小。但当λ过于大的时候，模型处于欠拟合，测试集RSS开始增加。

（c）问题（略）

iv.稳定减小。在λ为0的时候，有着一定的方差，当λ开始增加，模型预测的系数为常数，方差减小。

（d）问题（略）

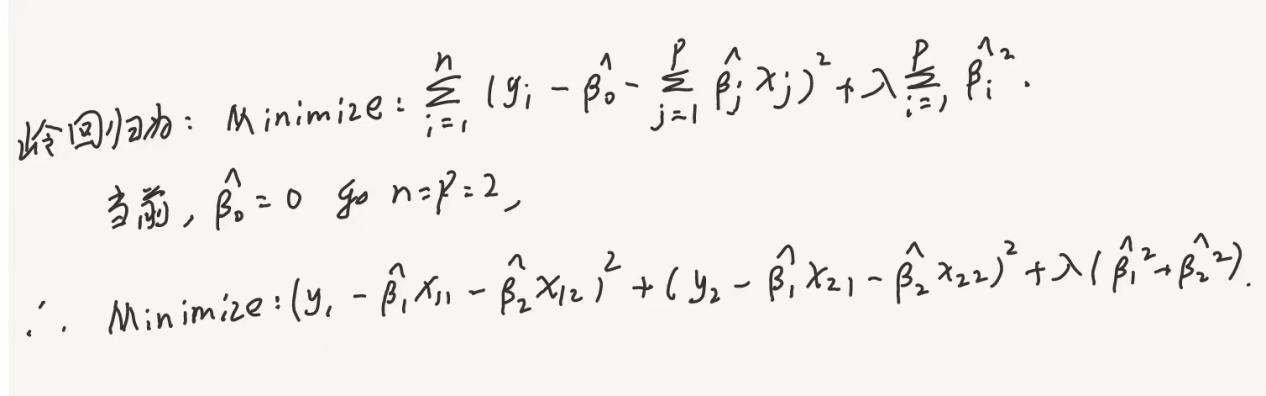
iii.稳定增加。在λ为0的时候，系数是最小二乘的结果，有着最小的偏差，而随着λ的增加，拟合效果越来越差，偏差便越来越大。

（e）问题（略）

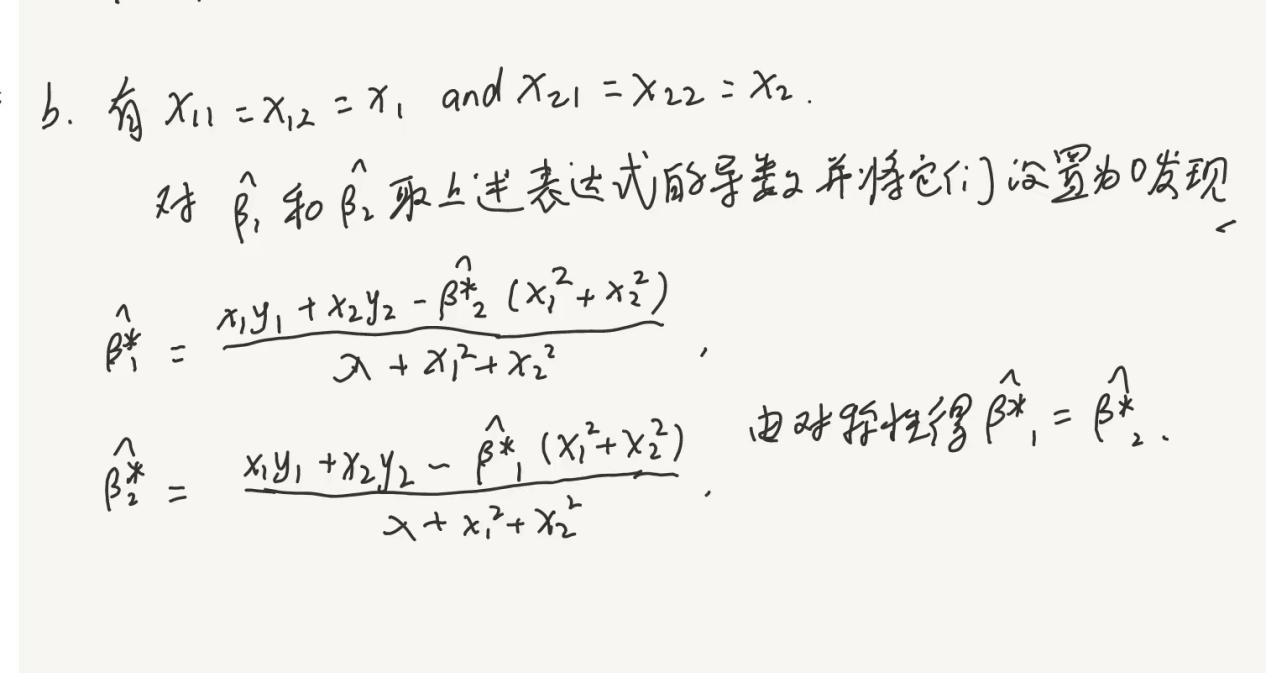
v.保持不变。根据不可约误差的定义，其与模型无关，因此λ的值的变化不会引起不可约误差的变化，所以其保持不变。

5.问题（略）

（a）问题（略）

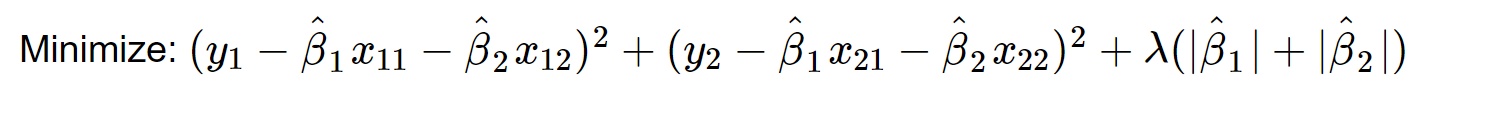


（b）问题（略）



（c）问题（略）

像岭回归一样，



（d）问题（略）

这是上面 c 中方程解的几何解释。我们使用 Lasso 约束的替代形式 |β^1|+|β^2|<s。

Lasso 约束采用 |β^1|+|β^2|<s 的形式，在绘制时采用以原点 (0,0) 为中心的菱形的熟悉形状。接下来考虑平方优化约束 (y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2。我们使用事实 x11=x12, x21=x22, x11+x21=0, x12+x22=0 和 y1+y2=0 将其简化为

最小化：2.(y1−(β^1+β^2)x11)2。

这个优化问题有一个简单的解：β^1+β^2=y1x11。这是一条平行于 Lasso-diamond β^1+β^2=s 边缘的线。现在原始 Lasso 优化问题的解决方案是函数 (y1−(β^1+β^2)x11)2 的轮廓，它与 Lasso-diamond β^1+β^2=s 相接触。最后，由于 β^1 和 β^2 非常沿着线 β^1+β^2=y1x11，这些轮廓在不同点接触 Lasso-diamond 边缘 β^1+β^2=s。因此，整条边 β^1+β^2=s 是 Lasso 优化问题的潜在解决方案！

可以对相反的 Lasso-菱形边进行类似的论证：β^1+β^2=-s。

因此，Lasso 问题没有唯一解。解的一般形式由两条线段给出：

β^1+β^2=s;β^1≥0;β^2≥0且β^1+β^2=-s;β^1≤0;β^2≤0

6.问题（略）

（a）问题（略）

y = 5

lambda = 5

betas = seq(-10, 10, 0.1)

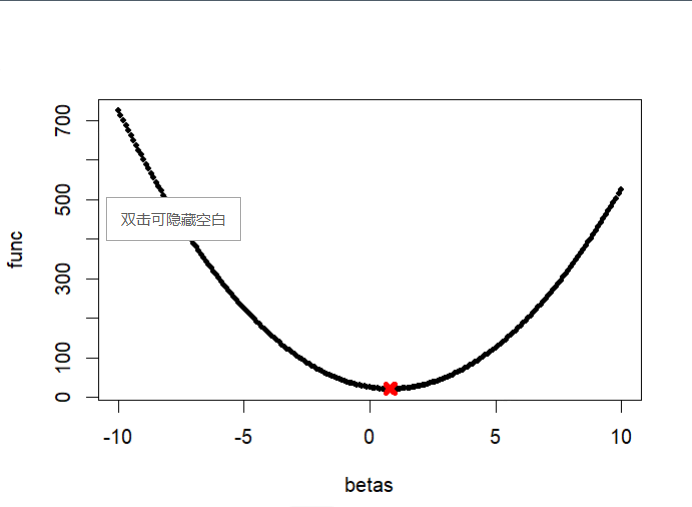
func = (y - betas)^2 + lambda \* betas^2

plot(betas, func, pch = 20,)

betaleast = y/(1 + lambda)

funcleast = (y - betaleast)^2 + lambda \* betaleast^2

points(betaleast, funcleast, col = "red", pch = 4, lwd = 5, cex = 1)



如图所示，6.14标出的红点为使6.12最小的点。

（b）问题（略）

y = 5

lambda = 5

betas = seq(-10, 10, 0.1)

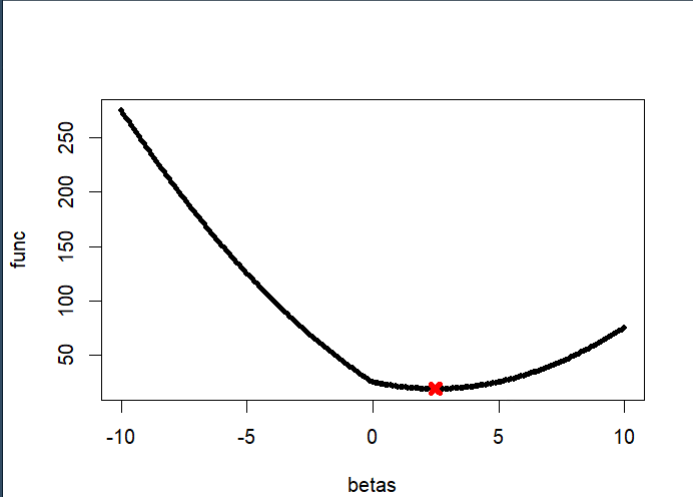
func = (y - betas)^2 + lambda \* abs(betas)

plot(betas, func, pch = 20,)

betaleast = y-lambda/2

funcleast = (y - betaleast)^2 + lambda \* abs(betaleast)

points(betaleast, funcleast, col = "red", pch = 4, lwd = 5, cex = 1)



如图所示，6.15标出的红点为使6.13最小的点。

**应用：**

8.问题（略）

（a）问题（略）

set.seed(1)

X = rnorm(100)

eps = rnorm(100)

（b）问题（略）

beta0 = 3

beta1 = 2

beta2 = -3

beta3 = 0.3

Y = beta0 + beta1 \* X + beta2 \* X^2 + beta3 \* X^3 + eps

（c）问题（略）

install.packages("leaps")

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)

which.min(mod.summary$cp)



which.min(mod.summary$bic)

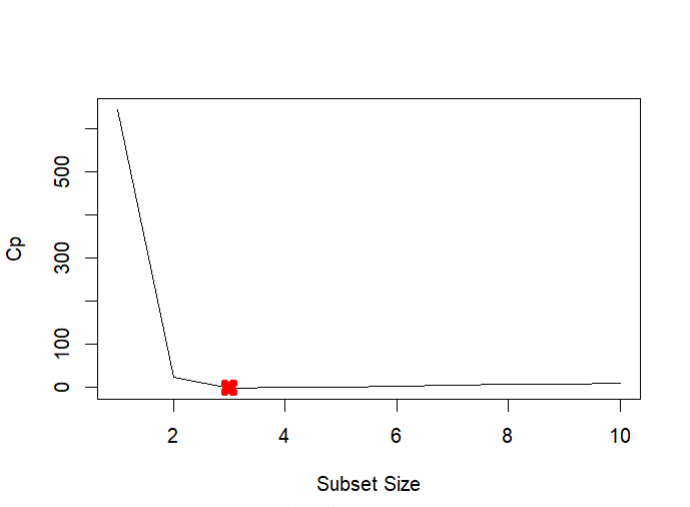


which.max(mod.summary$adjr2)



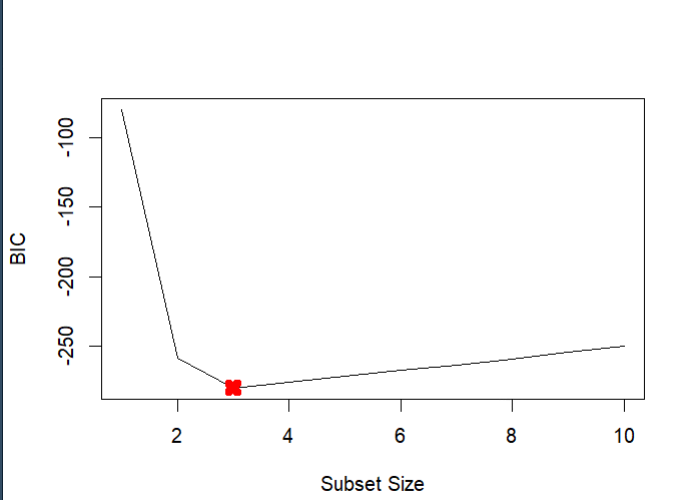
plot(mod.summary$cp, xlab = "Subset Size", ylab = "Cp", pch = 20, type = "l")

points(3, mod.summary$cp[3], pch = 4, col = "red", lwd = 7)



plot(mod.summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l")

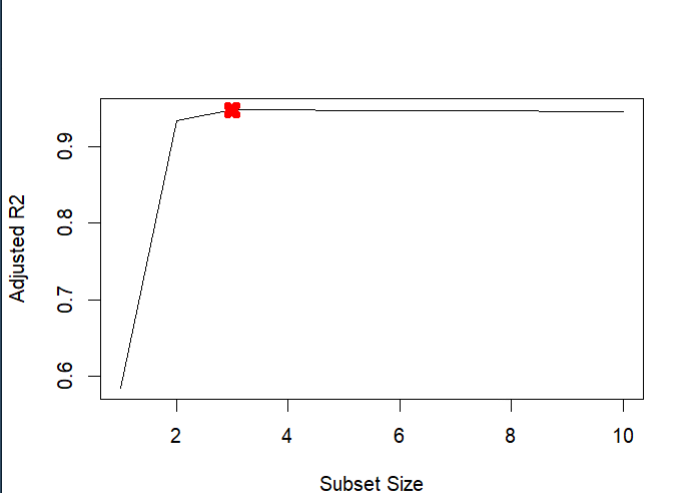
points(3, mod.summary$bic[3], pch = 4, col = "red", lwd = 7)



plot(mod.summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R2", pch = 20,

type = "l")

points(3, mod.summary$adjr2[3], pch = 4, col = "red", lwd = 7)



由结果可得，3，4变量模型有相同的Cp和adjr2。但3变量模型有着更小的BIC，因此将3变量模型选择为最优模型。

coefficients(mod.full, id = 3)



得到的系数与设定的系数相差不大。

（d）问题（略）

向前逐步选择：

mod.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,

method = "forward")

mod.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,

method = "backward")

fwd.summary = summary(mod.fwd)

bwd.summary = summary(mod.bwd)

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)



par(mfrow = c(3, 2))

plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward Cp", pch = 20, type = "l")

points(3, fwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward Cp", pch = 20, type = "l")

points(3, bwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(fwd.summary$bic, xlab = "Subset Size", ylab = "Forward BIC", pch = 20,

type = "l")

points(3, fwd.summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$bic, xlab = "Subset Size", ylab = "Backward BIC", pch = 20,

type = "l")

points(3, bwd.summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R2",

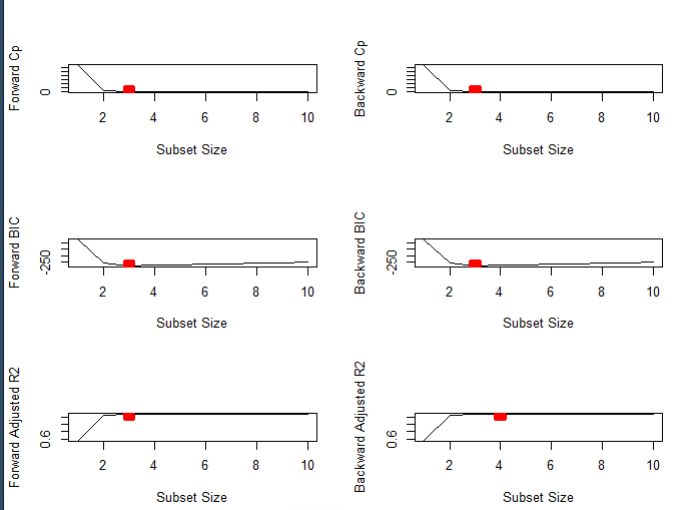
pch = 20, type = "l")

points(3, fwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R2",

pch = 20, type = "l")

points(4, bwd.summary$adjr2[4], pch = 4, col = "red", lwd = 7)



我们看到所有统计数据都选择了3个变量模型，除了反向逐步调整R2。下面是系数：

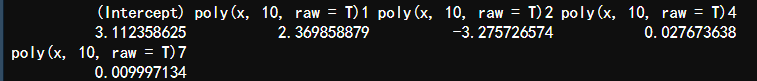
coefficients(mod.fwd, id = 3)



coefficients(mod.bwd, id = 3)



coefficients(mod.fwd, id = 4)



这里逐步选择X^7 / X^3。3个变量的逐步后退选择X^9, 4个变量的逐步后退选择X^4和X^7。所有其他系数都接近β s。

（e）问题（略）

install.packages(“glmnet”)

xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]

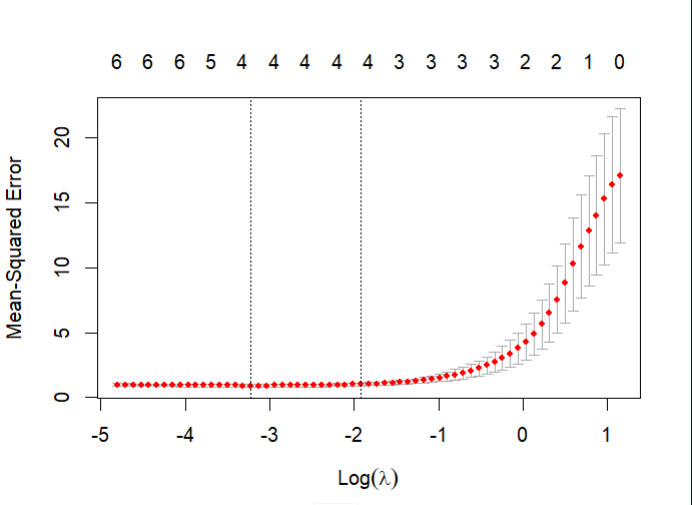
mod.lasso = cv.glmnet(xmat, Y, alpha = 1)

best.lambda = mod.lasso$lambda.min

best.lambda

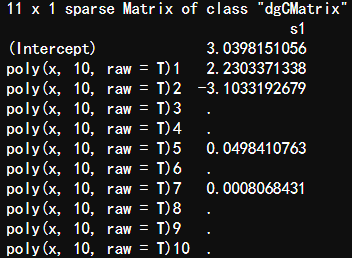


plot(mod.lasso)



best.model = glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type = "coefficients")



由结果可得，常数项，一次项，二次项的系数与设定系数很接近。

虽然出现了五次项，七次项，由于值很小，可以忽略不计。

（f）问题（略）

beta7 = 7

Y = beta0 + beta7 \* X^7 + eps

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)

which.min(mod.summary$cp)



which.min(mod.summary$bic)



which.max(mod.summary$adjr2)



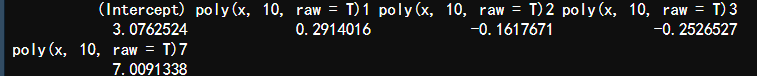
coefficients(mod.full, id = 1)



coefficients(mod.full, id = 2)



coefficients(mod.full, id = 4)



我们看到，BIC选择了具有匹配系数的最精确的1-变量模型。其他标准选择了额外的变量。

xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]

mod.lasso = cv.glmnet(xmat, Y, alpha = 1)

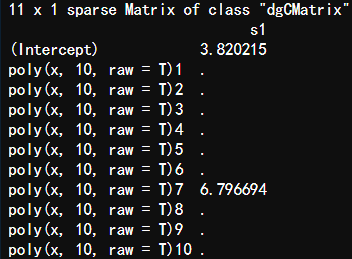
best.lambda = mod.lasso$lambda.min

best.lambda



best.model = glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type = "coefficients")



Lasso也选择了最好的单变量模型，但是intercept非常差(3.8 vs 3)。

9.问题（略）

（a）问题（略）

library(ISLR)

set.seed(11)

sum(is.na(College))



train.size = dim(College)[1] / 2

train = sample(1:dim(College)[1], train.size)

test = -train

College.train = College[train, ]

College.test = College[test, ]

（b）问题（略）

lm.fit = lm(Apps~., data=College.train)

lm.pred = predict(lm.fit, College.test)

mean((College.test[, "Apps"] - lm.pred)^2)



测试误差为1026096

（c）问题（略）

library(glmnet)

train.mat = model.matrix(Apps~., data=College.train)

test.mat = model.matrix(Apps~., data=College.test)

grid = 10 ^ seq(4, -2, length=100)

mod.ridge = cv.glmnet(train.mat, College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)

lambda.best = mod.ridge$lambda.min

lambda.best



ridge.pred = predict(mod.ridge, newx=test.mat, s=lambda.best)

mean((College.test[, "Apps"] - ridge.pred)^2)



测试误差比OLS稍微小一点，是1026069.

（d）问题（略）

mod.lasso = cv.glmnet(train.mat, College.train[, "Apps"], alpha=1, lambda=grid, thresh=1e-12)

lambda.best = mod.lasso$lambda.min

lambda.best



lasso.pred = predict(mod.lasso, newx=test.mat, s=lambda.best)

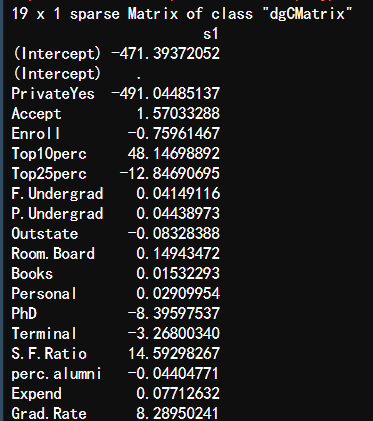
mean((College.test[, "Apps"] - lasso.pred)^2)



测试误差稍微比OLS小一点，是1026036.

mod.lasso = glmnet(model.matrix(Apps~., data=College), College[, "Apps"], alpha=1)

predict(mod.lasso, s=lambda.best, type="coefficients")



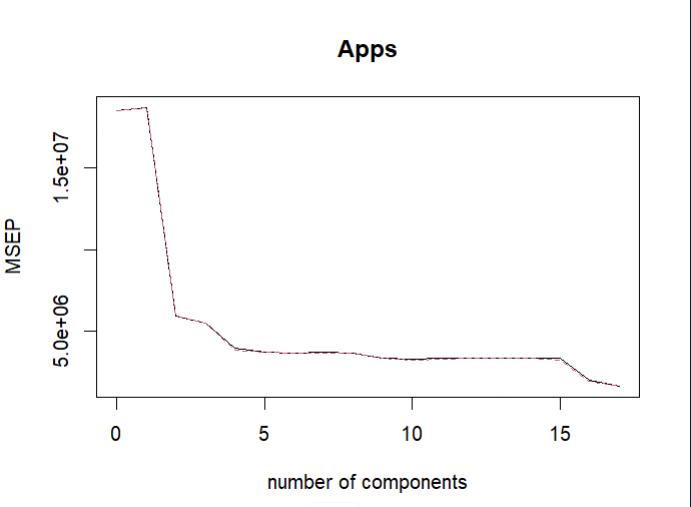
（e）问题（略）

Install.packages(“pls”)

Libary(pls)

pcr.fit = pcr(Apps~., data=College.train, scale=T, validation="CV")

validationplot(pcr.fit, val.type="MSEP")



取M为10，

pcr.pred = predict(pcr.fit, College.test, ncomp=10)

mean((College.test[, "Apps"] - pcr.pred)^2)

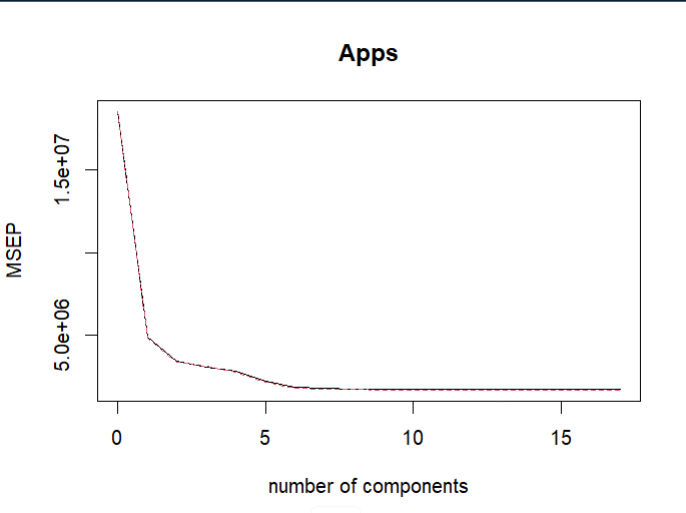


测试误差为1867486.

（f）问题（略）

pls.fit = plsr(Apps~., data=College.train, scale=T, validation="CV")

validationplot(pls.fit, val.type="MSEP")



pls.pred = predict(pls.fit, College.test, ncomp=10)

mean((College.test[, "Apps"] - pls.pred)^2)



测试误差为1031287

（g）问题（略）

求出每一个模型的测试R方：

lm.test.r2 = 1 - mean((College.test[, "Apps"] - lm.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)



ridge.test.r2 = 1 - mean((College.test[, "Apps"] - ridge.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)



lasso.test.r2 = 1 - mean((College.test[, "Apps"] - lasso.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)



pcr.test.r2 = 1 - mean((College.test[, "Apps"] - pcr.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)



pls.test.r2 = 1 - mean((College.test[, "Apps"] - pls.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)



根据结果所示，最小二乘，岭回归，lasso和PLS模型的测试R方都在0.9左右，只有PCR的测试R方低于0.9。

所以最小二乘，岭回归，lasso和PLS模型可以用来较为准确地预测申请人数。

10.问题（略）

（a）问题（略）

set.seed(1)

p = 20

n = 1000

x = matrix(rnorm(n \* p), n, p)

B = rnorm(p)

B[3] = 0

B[4] = 0

B[9] = 0

B[19] = 0

B[10] = 0

eps = rnorm(p)

y = x %\*% B + eps

（b）问题（略）

train = sample(seq(1000), 100, replace = FALSE)

y.train = y[train, ]

y.test = y[-train, ]

x.train = x[train, ]

x.test = x[-train, ]

（c）问题（略）

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)

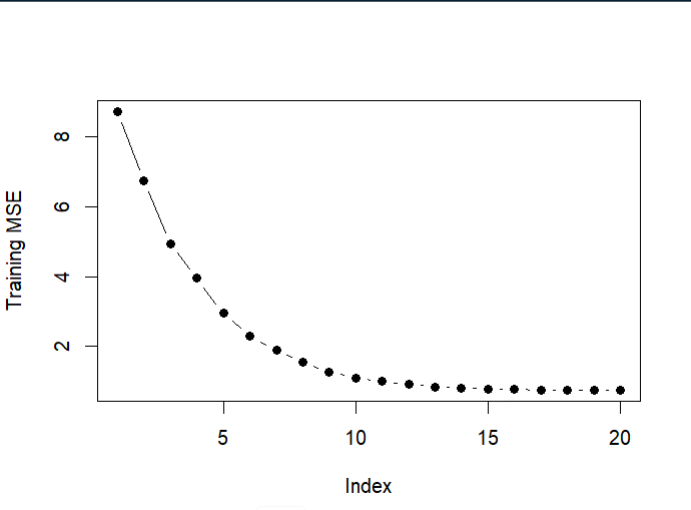
pred = as.matrix(x.train[, x\_cols %in% names(coefi)]) %\*% coefi[names(coefi) %in%

x\_cols]

val.errors[i] = mean((y.train - pred)^2)

}

plot(val.errors, ylab = "Training MSE", pch = 19, type = "b")



（d）问题（略）

val.errors = rep(NA, p)

for (i in 1:p) {

coefi = coef(regfit.full, id = i)

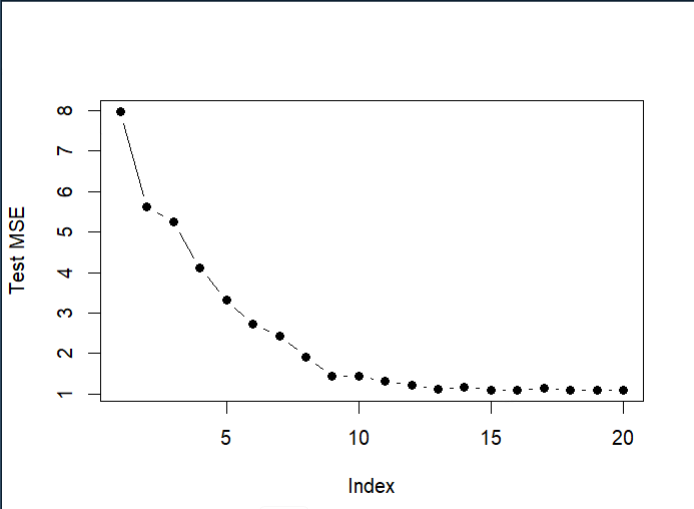
pred = as.matrix(x.test[, x\_cols %in% names(coefi)]) %\*% coefi[names(coefi) %in%

x\_cols]

val.errors[i] = mean((y.test - pred)^2)

}

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



（e）问题（略）

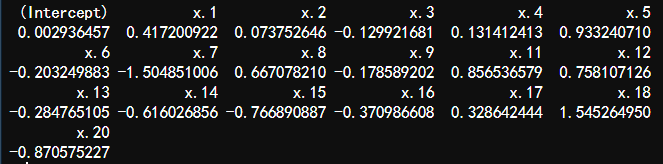
which.min(val.errors)



在含有18个特征的时候，测试MSE取值最小。因为在a问中设置了5个b中的元素为0。所以在有18个特征左右的时候，测试MSE应该最小，如上图所示，在特征数为18时，测试MSE最小。

（f）问题（略）

coef(regfit.full, id = 18)



设置的B元素为0中，10、19成功没有被捕捉到，3、4、9错误得被捕捉到了。但与实际模型已经有着较高的拟合度。

（g）问题（略）

val.errors = rep(NA, p)

a = rep(NA, p)

b = rep(NA, p)

for (i in 1:p) {

coefi = coef(regfit.full, id = i)

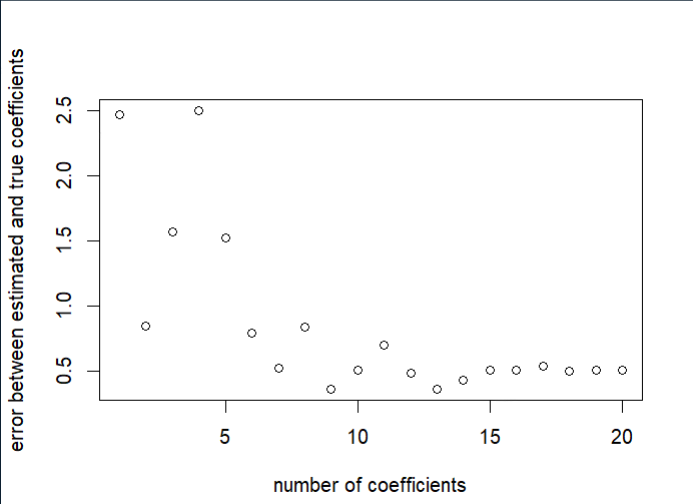
a[i] = length(coefi) - 1

b[i] = sqrt(sum((B[x\_cols %in% names(coefi)] - coefi[names(coefi) %in% x\_cols])^2) +

sum(B[!(x\_cols %in% names(coefi))])^2)

}

plot(x = a, y = b, xlab = "number of coefficients", ylab = "error between estimated and true coefficients")



which.min(b)



与d得到的图像对比，可以看出两个图像较为相似，虽然求出的使error最小的变量个数为9，但从图像中可以看到与18差距在可接受的范围内。