高级统计方法 第7次作业:

序号：27 姓名：王琪瑞 学号：20202241014 班级：软2002

**概念**

3.问题

（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。

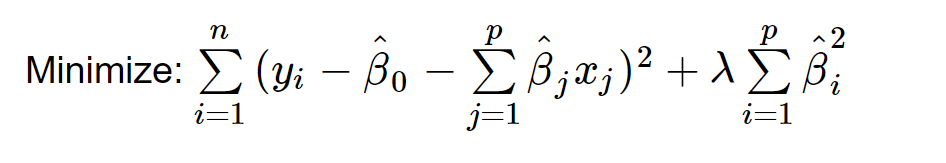
（e）问题（略）

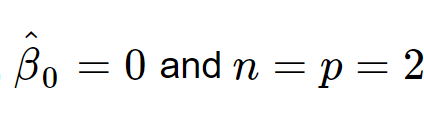
v.保持不变。根据不可约误差的定义，其与模型无关，因此λ的值的变化不会引起不可约误差的变化，所以其保持不变。

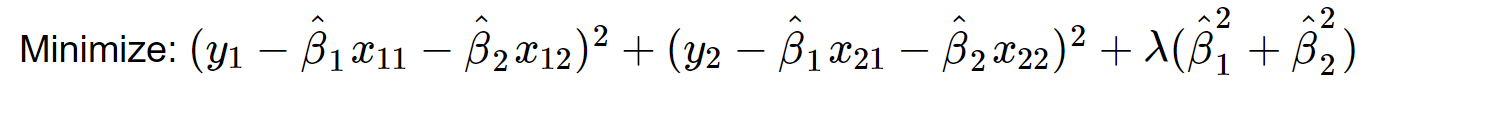
5.问题（略）

a.

岭回归为:

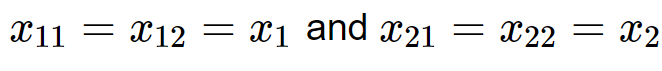


当前，

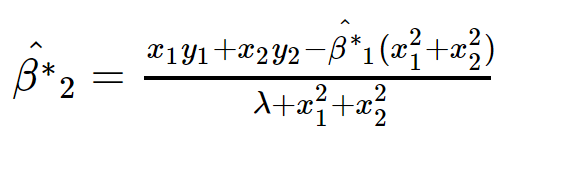
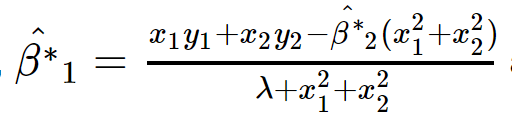
所以

b.

我们有



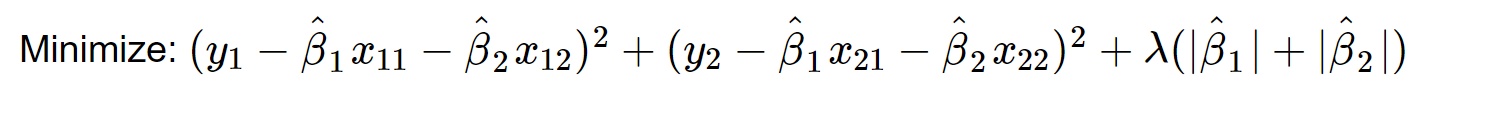
我们对 β1^ 和 β2^ 取上述表达式的导数并将它们设置为零发现，



这些表达式的对称性表明 β∗^1=β∗^2

c.

像岭回归一样，



d.

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）

作图脚本：

y = 5

lambda = 5

betas = seq(-10, 10, 0.1)

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

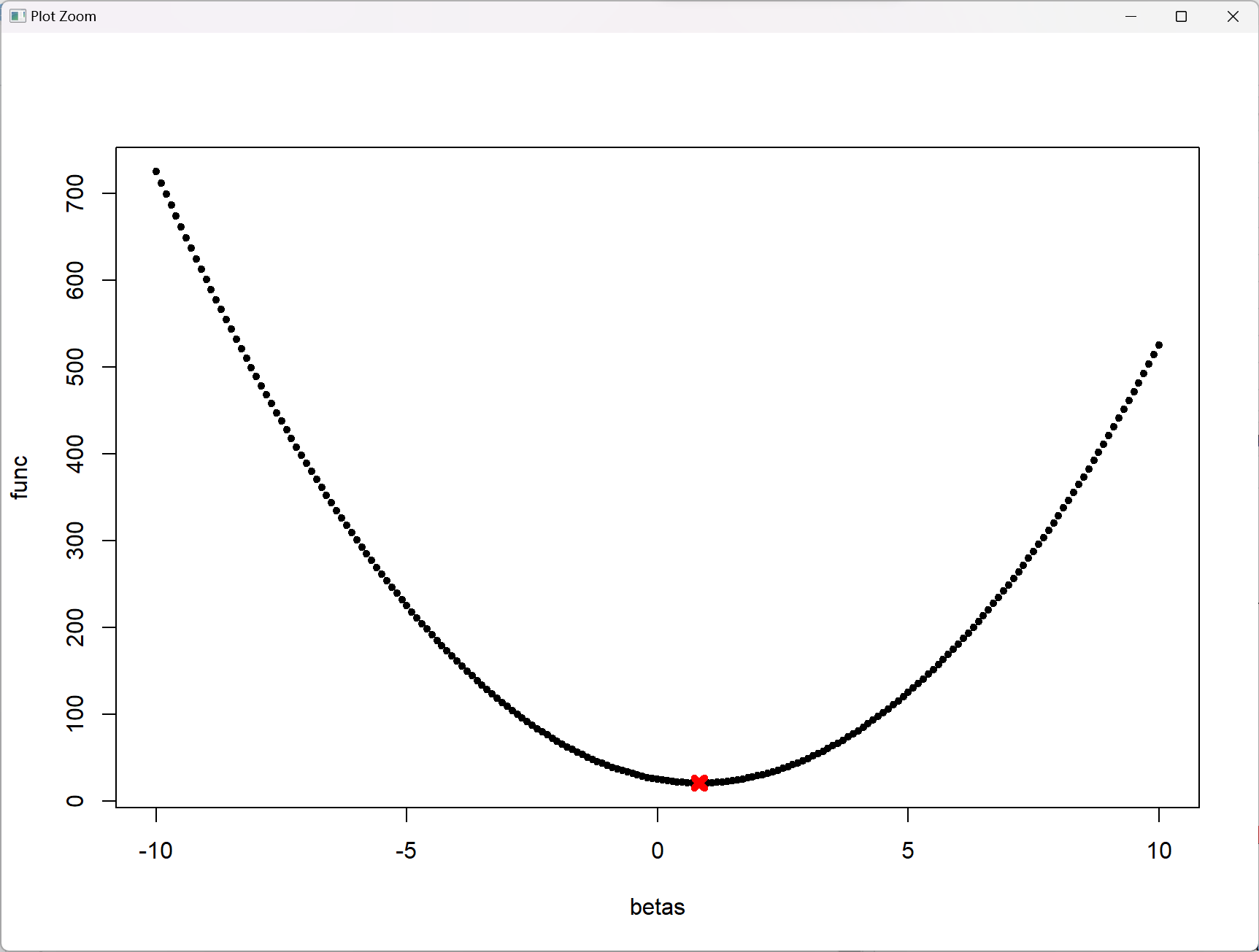
plot(betas, func, pch = 20,)

betaleast = y/(1 + lambda)

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

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

图像：



如图所示，6.14标出的红点为使6.12最小的点。

（b）问题（略）

作图脚本：

y = 5

lambda = 5

betas = seq(-10, 10, 0.1)

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

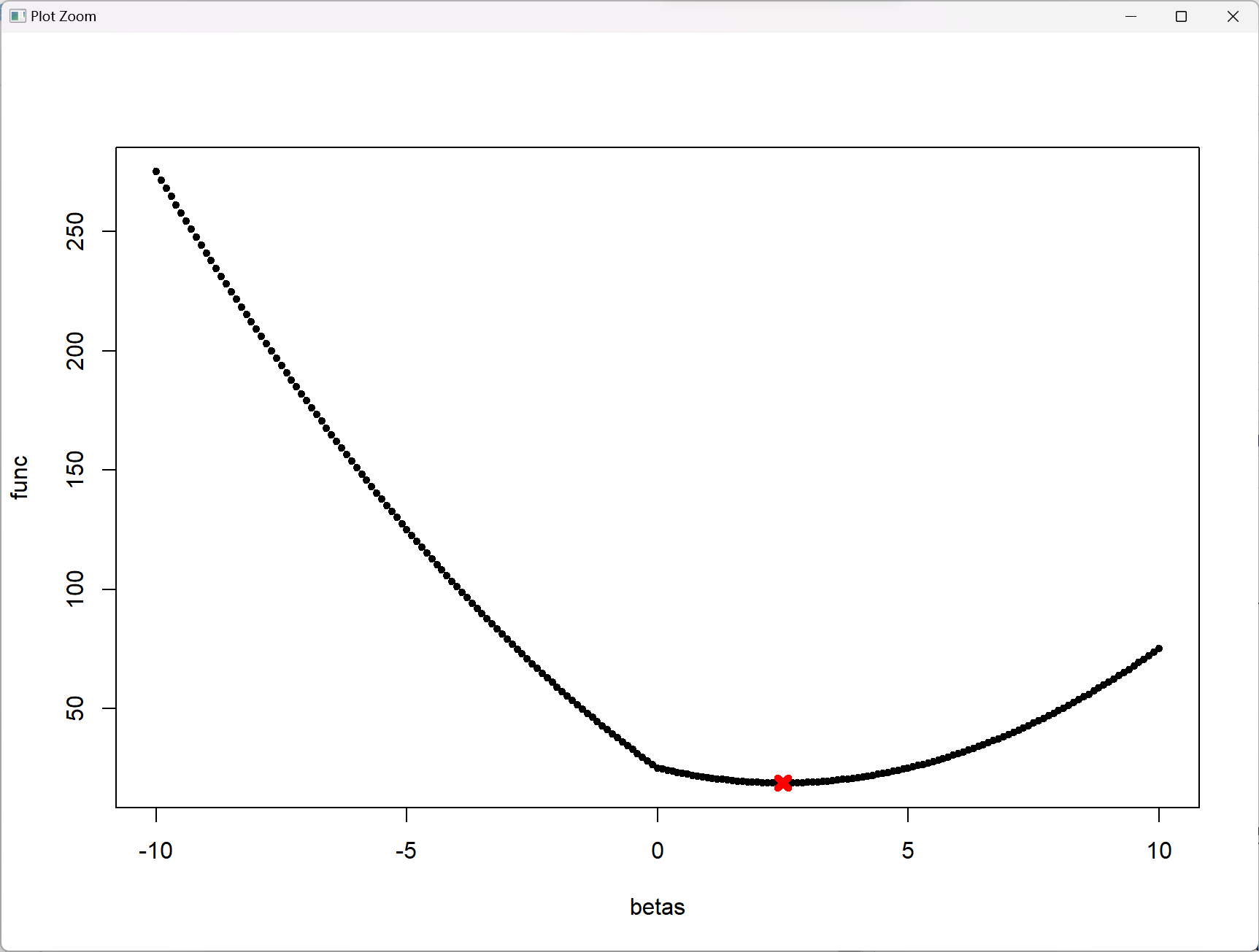
plot(betas, func, pch = 20,)

betaleast = y-lambda/2

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

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

图像：



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）

脚本：

set.seed(1)

X = rnorm(100)

eps = rnorm(100)

（b）问题（略）

脚本：

Y =10 -3 \* X + 3 \* X^2 -3 \* X^3 + eps

（c）问题（略）

脚本：

library(leaps)

Data = data.frame(y = Y, x = X)

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

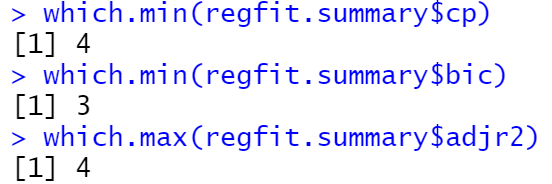
regfit.summary = summary(regfit.full)

which.min(regfit.summary$cp)

which.min(regfit.summary$bic)

which.max(regfit.summary$adjr2)

截图：



根据Cp和调整后R方选择出的最佳模型是4变量模型。

根据BIC选择出的最佳模型是3变量模型。

脚本：

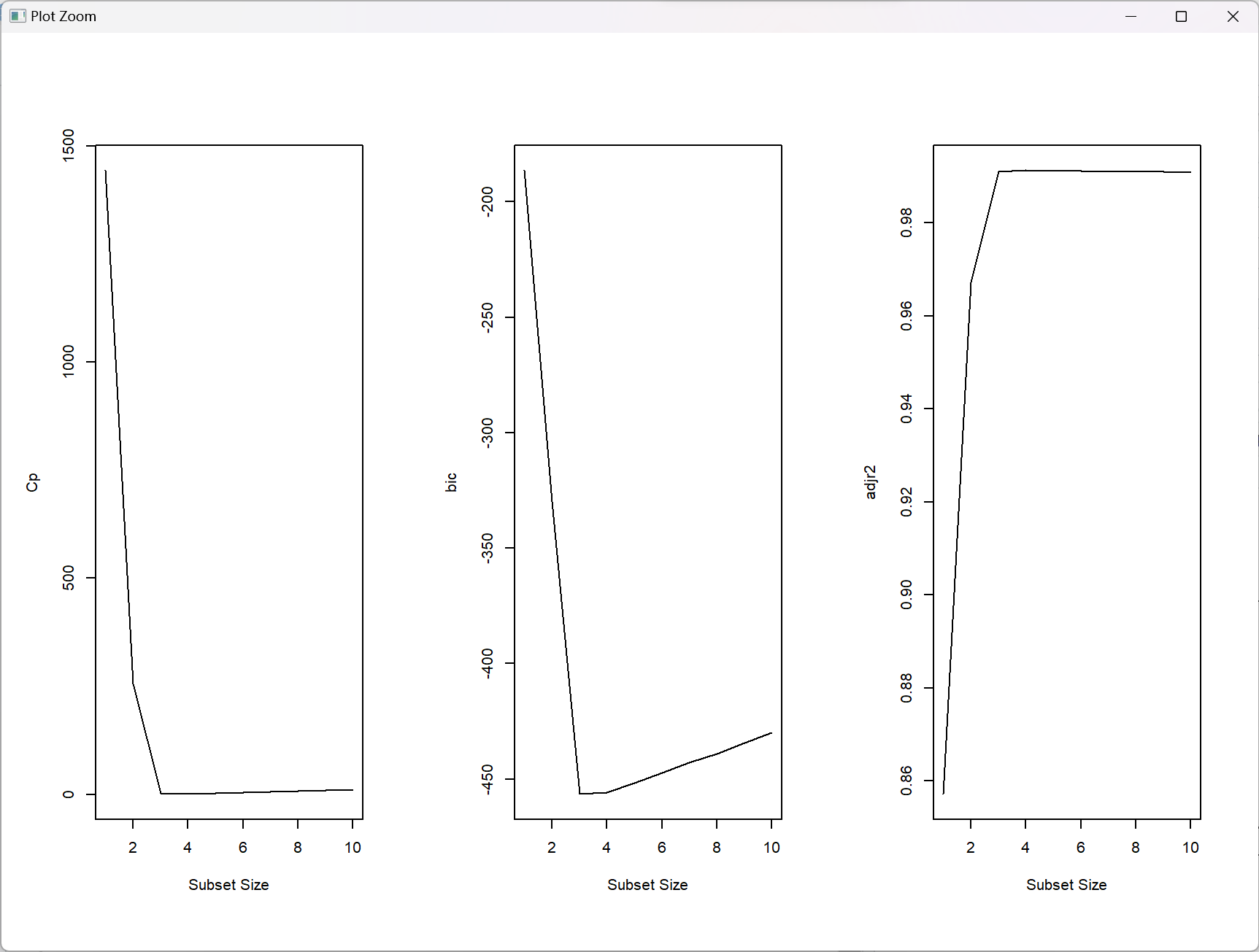
par(mfrow=c(1,3))

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

plot(regfit.summary$bic, xlab = "Subset Size", ylab = "bic", type = "l")

plot(regfit.summary$adjr2, xlab = "Subset Size", ylab = "adjr2", type = "l")

截图：

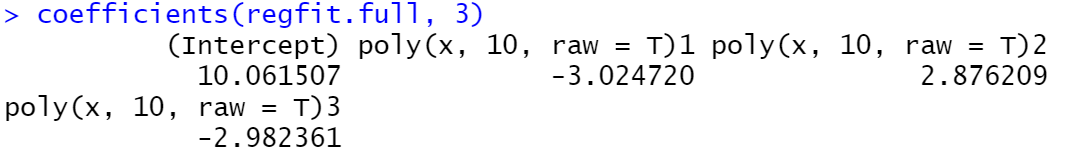


由结果可得，3，4变量模型有相同的Cp和adjr2。但3变量模型有着更小的BIC，因此将3变量模型选择为最优模型。

脚本：

coefficients(regfit.full, 3)

截图：



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：

脚本：

regfit.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = Data, nvmax = 10, method = "forward")

fwd.summary = summary(regfit.fwd)

which.min(fwd.summary$cp)

which.min(fwd.summary$bic)

which.max(fwd.summary$adjr2)

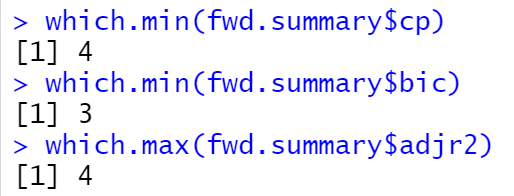
plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Cp", type = "l")

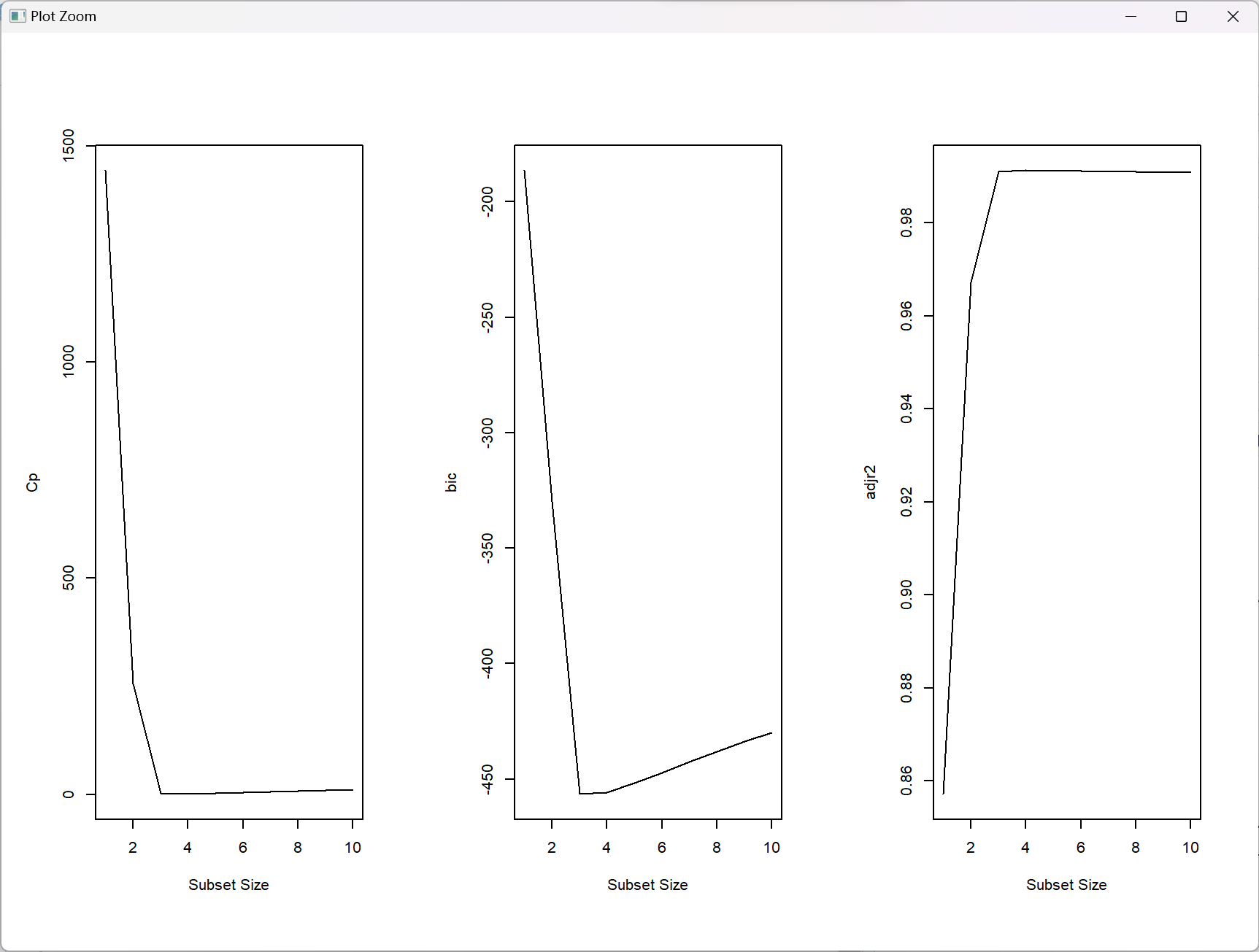
plot(fwd.summary$bic, xlab = "Subset Size", ylab = "bic", type = "l")

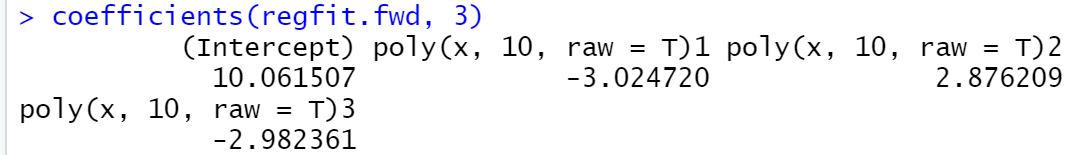
plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "adjr2", type = "l")

coefficients(regfit.fwd, 3)

截图：







向后逐步选择：

脚本：

regfit.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = Data, nvmax = 10, method = "backward")

bwd.summary = summary(regfit.bwd)

which.min(bwd.summary$cp)

which.min(bwd.summary$bic)

which.max(bwd.summary$adjr2)

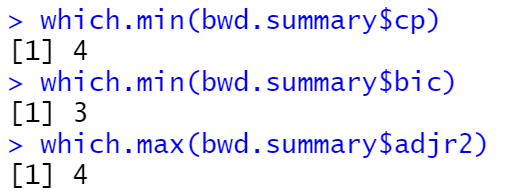
plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Cp", type = "l")

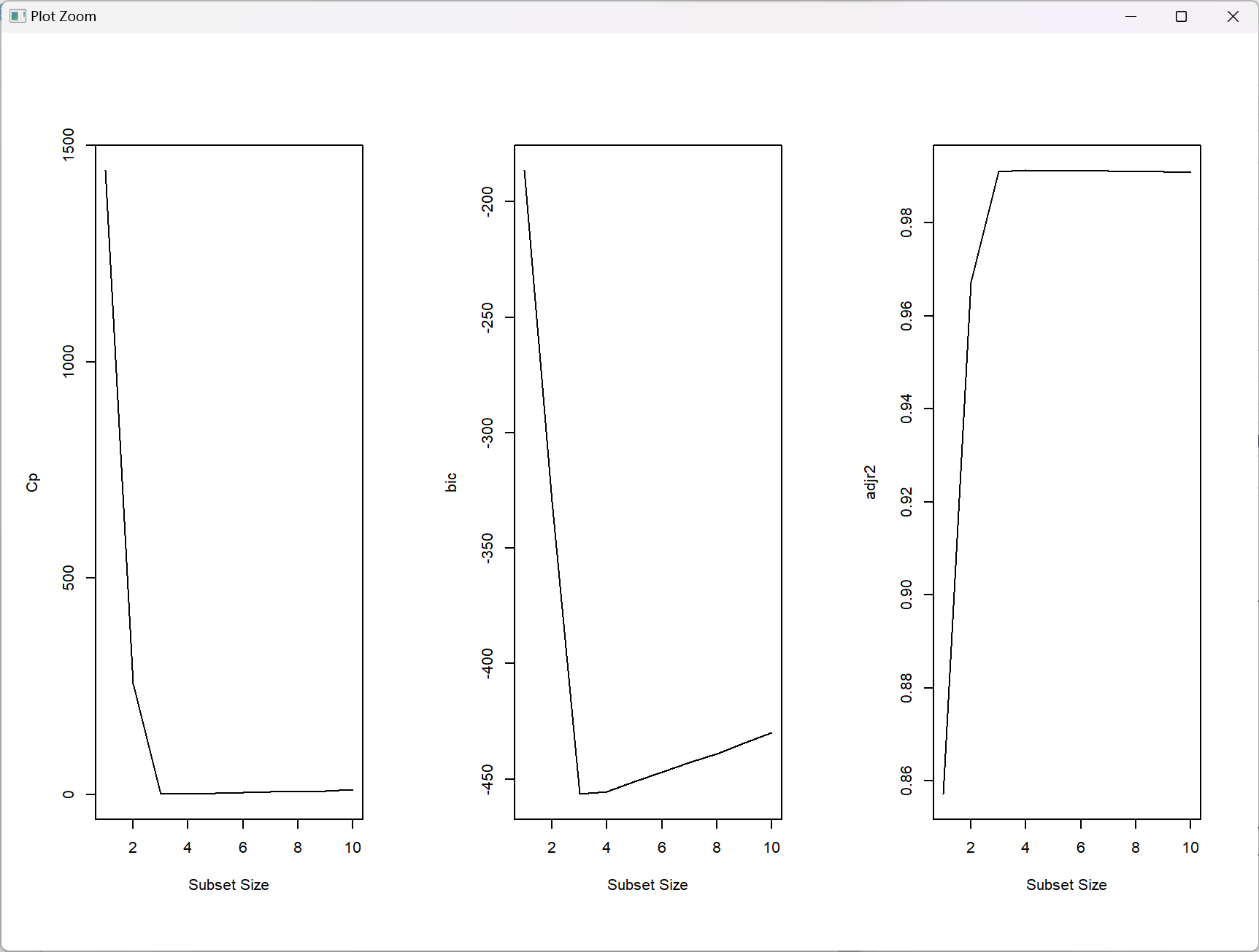
plot(bwd.summary$bic, xlab = "Subset Size", ylab = "bic", type = "l")

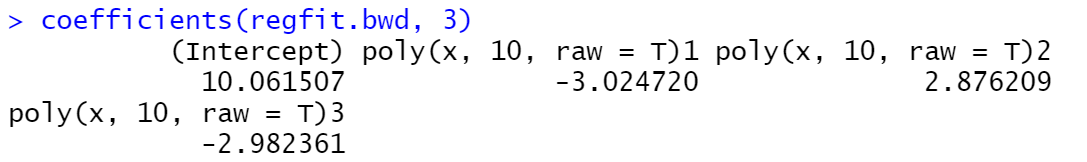
plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "adjr2", type = "l")

coefficients(regfit.bwd, 3)

截图：







由结果可知，向前和向后逐步选择选择的最佳模型都与最优子集选择相同。并且计算出的系数均与设定的系数相近。

（e）问题（略）

脚本：

library(glmnet)

lasso.mod = cv.glmnet(model.matrix(y ~ poly(x, 10, raw = T), data = Data)[, -1], Y, alpha = 1)

bestlam = lasso.mod$lambda.min

bestlam

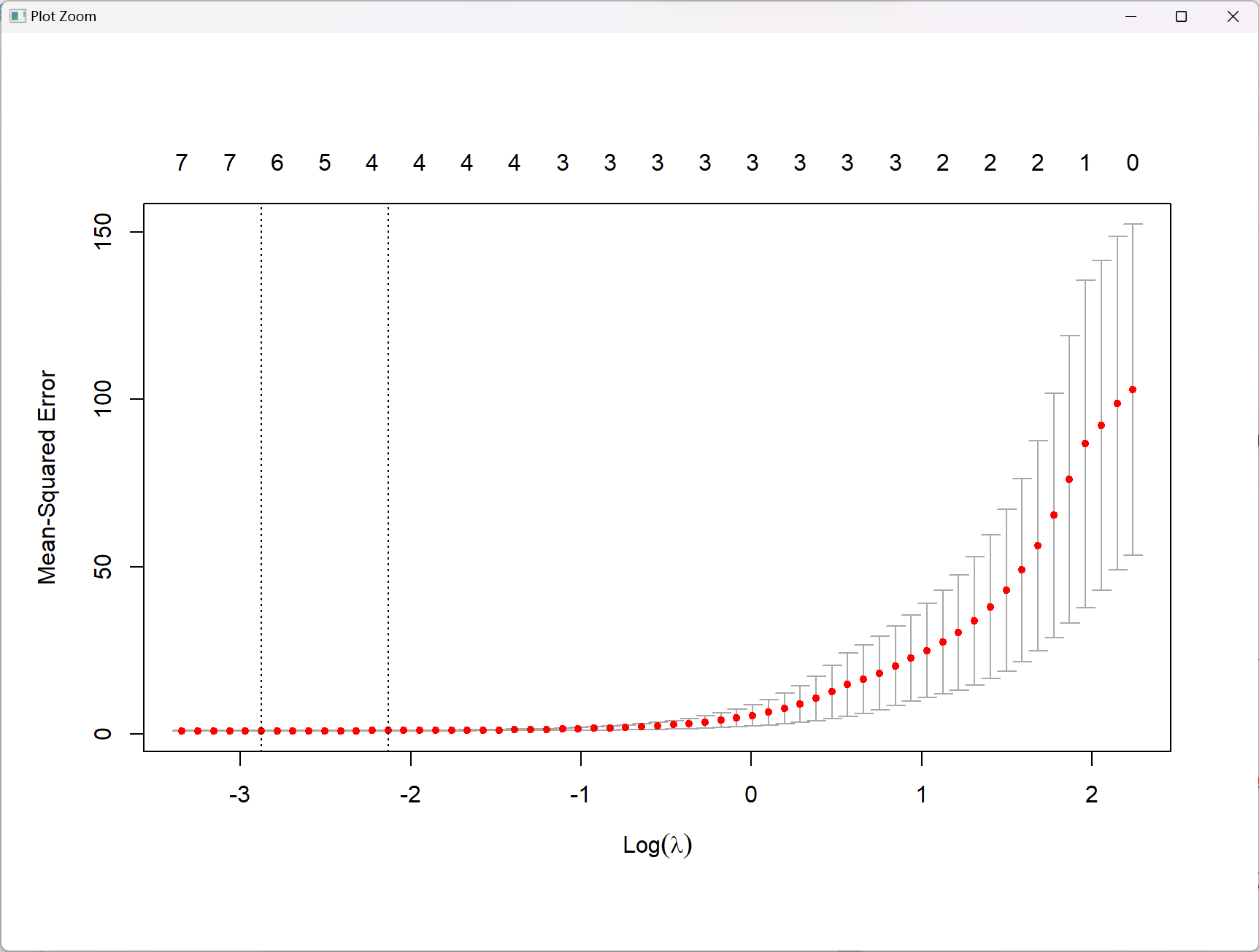
plot(lasso.mod)

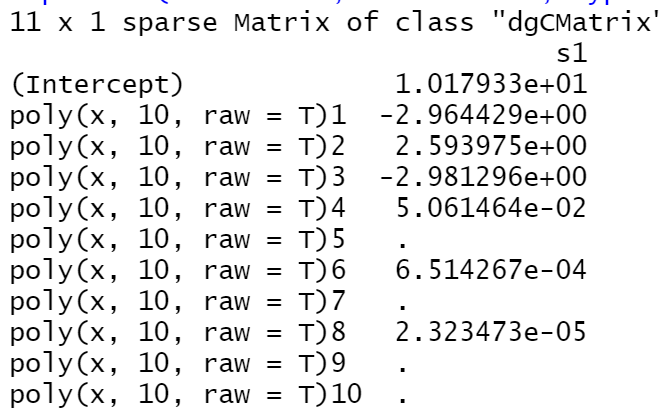
best.model = glmnet(model.matrix(y ~ poly(x, 10, raw = T), data = Data)[, -1], Y, alpha = 1)

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

截图：







由结果可得，常数项，一次项，二次项和三次项的系数与设定系数很接近。

虽然出现了四次项，六次项和八次项，由于值很小，可以忽略不计。

（f）问题（略）

脚本：

Y = 10 + 15 \* X^7 + eps

Data = data.frame(y = Y, x = X)

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

regfit.summary = summary(regfit.full)

which.min(regfit.summary$cp)

which.min(regfit.summary$bic)

which.max(regfit.summary$adjr2)

par(mfrow=c(1,3))

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

plot(regfit.summary$bic, xlab = "Subset Size", ylab = "bic", type = "l")

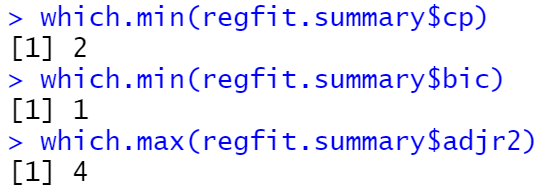
plot(regfit.summary$adjr2, xlab = "Subset Size", ylab = "adjr2", type = "l")

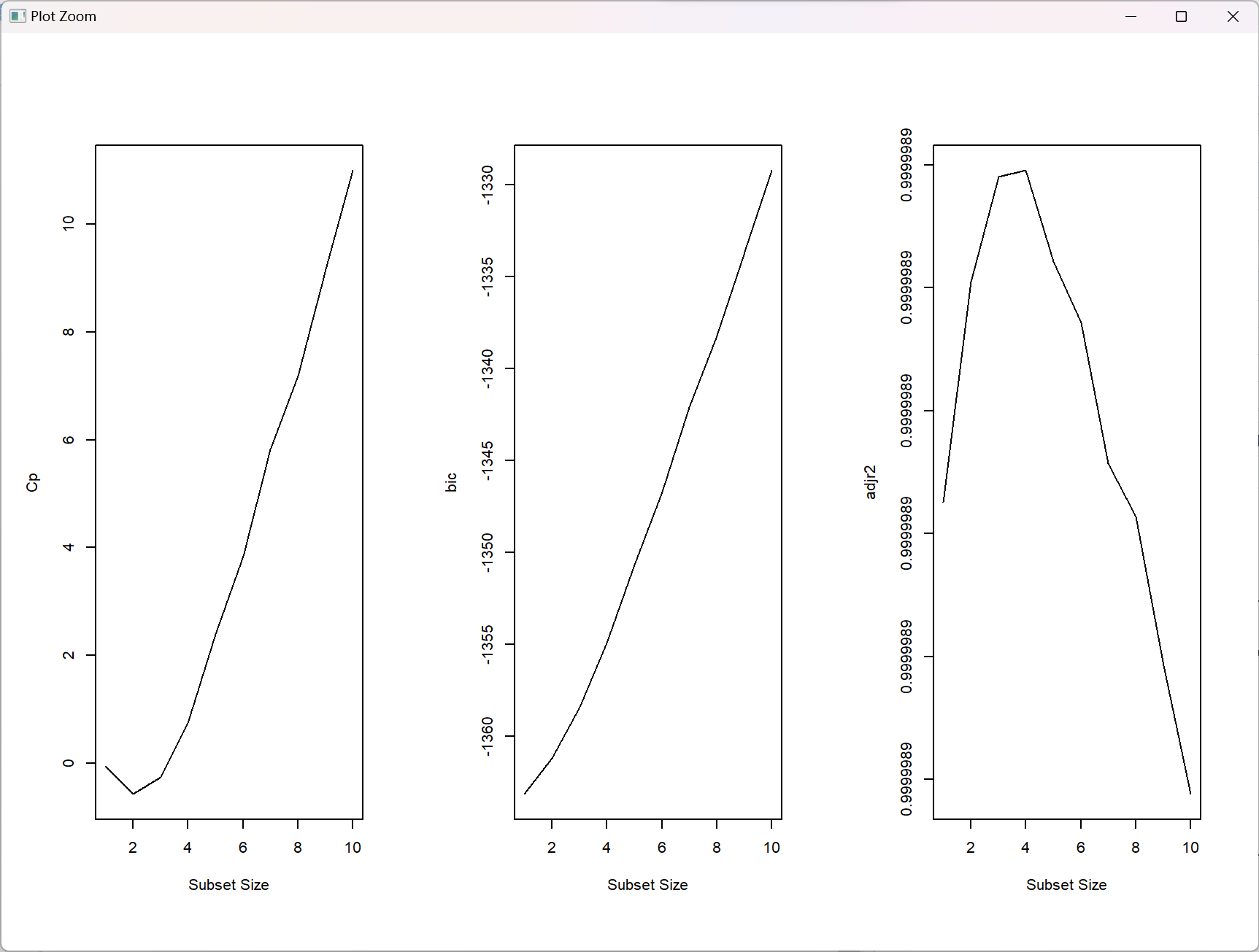
coefficients(regfit.full, 2)

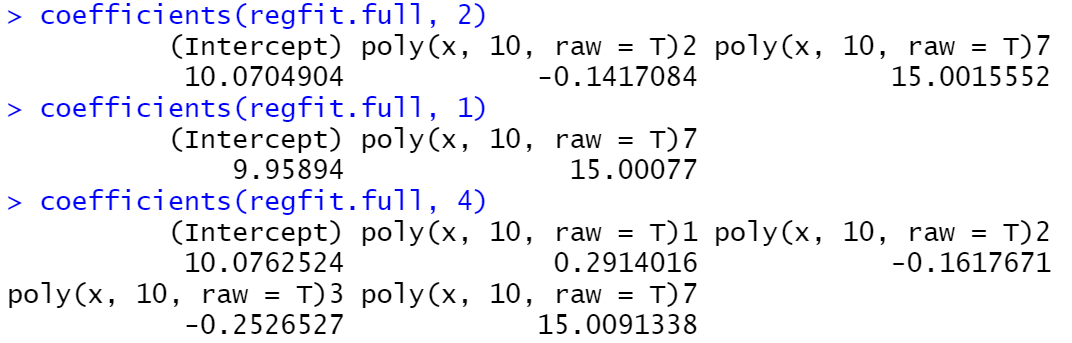
coefficients(regfit.full, 1)

coefficients(regfit.full, 4)

截图：







由结果看到，1变量模型可以选择为最优模型。

脚本：

lasso.mod = cv.glmnet(model.matrix(y ~ poly(x, 10, raw = T), data = Data)[, -1], Y, alpha = 1)

bestlam = lasso.mod$lambda.min

bestlam

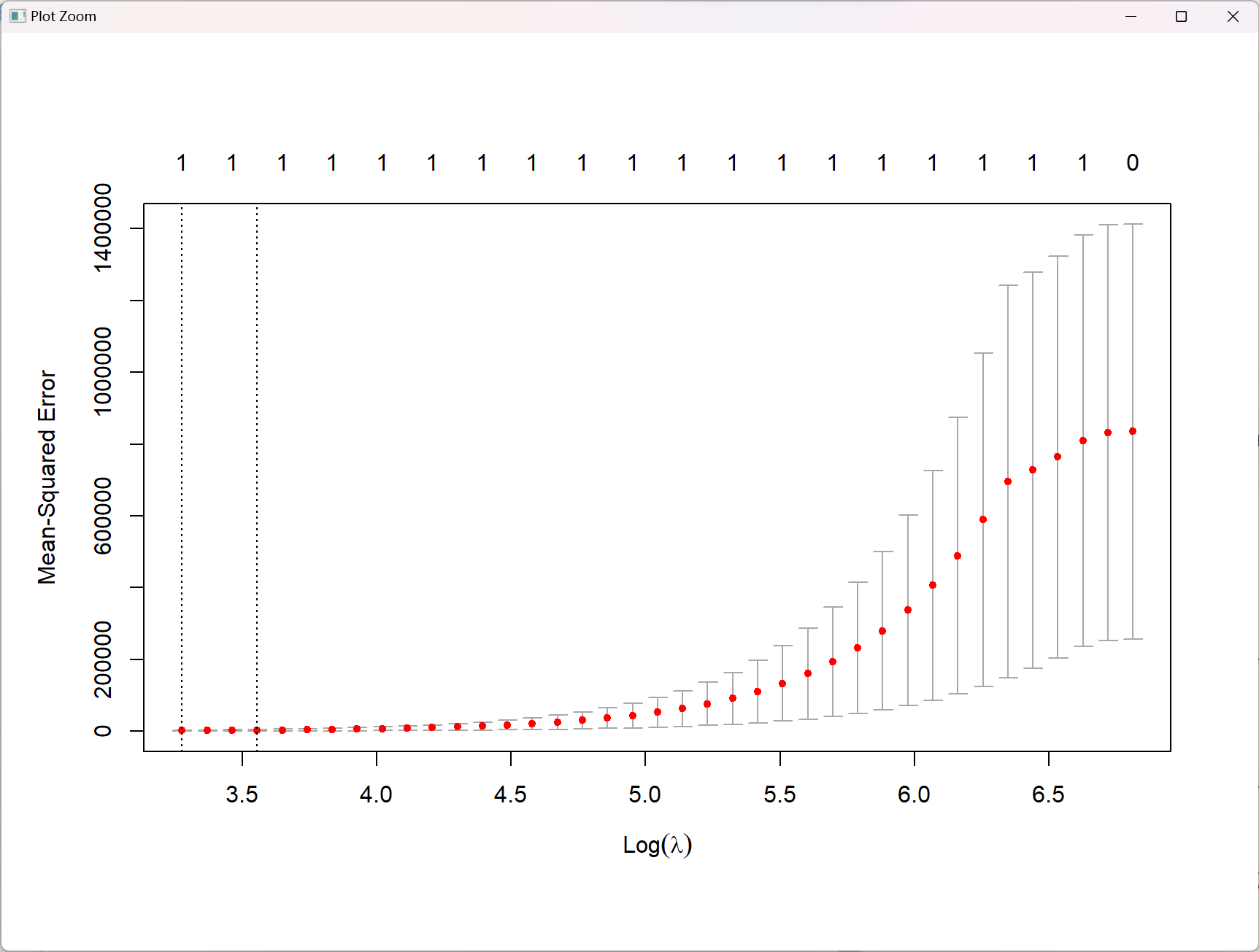
plot(lasso.mod)

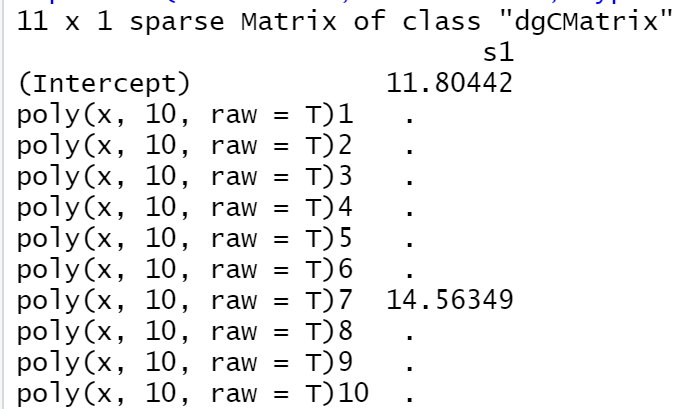
best.model = glmnet(model.matrix(y ~ poly(x, 10, raw = T), data = Data)[, -1], Y, alpha = 1)

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

截图：







由lasso结果可以得到，常数项和七次项的系数都与设定系数相近。

9.问题（略）

（a）问题（略）

脚本：

library(ISLR)

set.seed(1)

train = sample(1:dim(College)[1], dim(College)[1] / 2)

test = -train

College.train = College[train, ]

College.test = College[test, ]

（b）问题（略）

脚本:

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：



（c）问题（略）

脚本：

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)

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

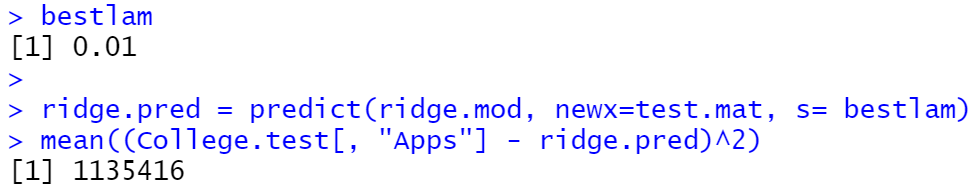
bestlam=ridge.mod$lambda.min

bestlam

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

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

截图:



测试误差略小一点。

（d）问题（略）

脚本：

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

bestlam = lasso.mod$lambda.min

bestlam

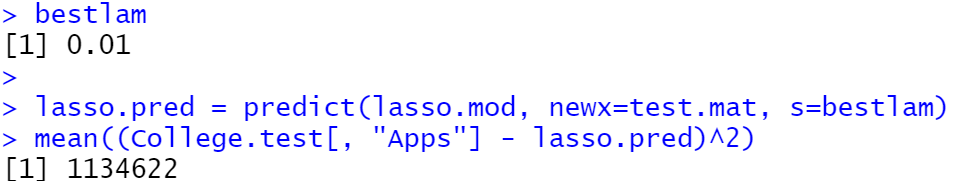
lasso.pred = predict(lasso.mod, newx=test.mat, s=bestlam)

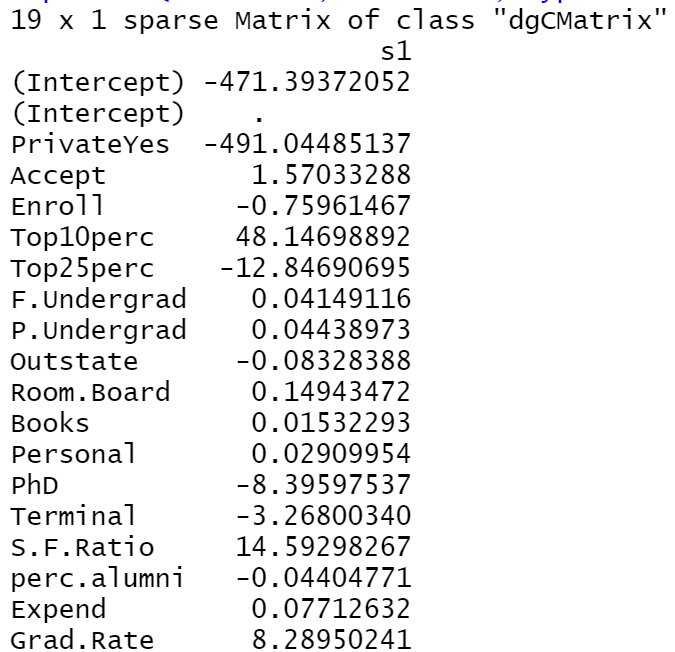
mean((College.test[, "Apps"] - lasso.pred)^2)

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

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

截图:





测试误差也略小一点。

系数估计值均非0。

将接近0认为非0，则非零系数有8个。

（e）问题（略）

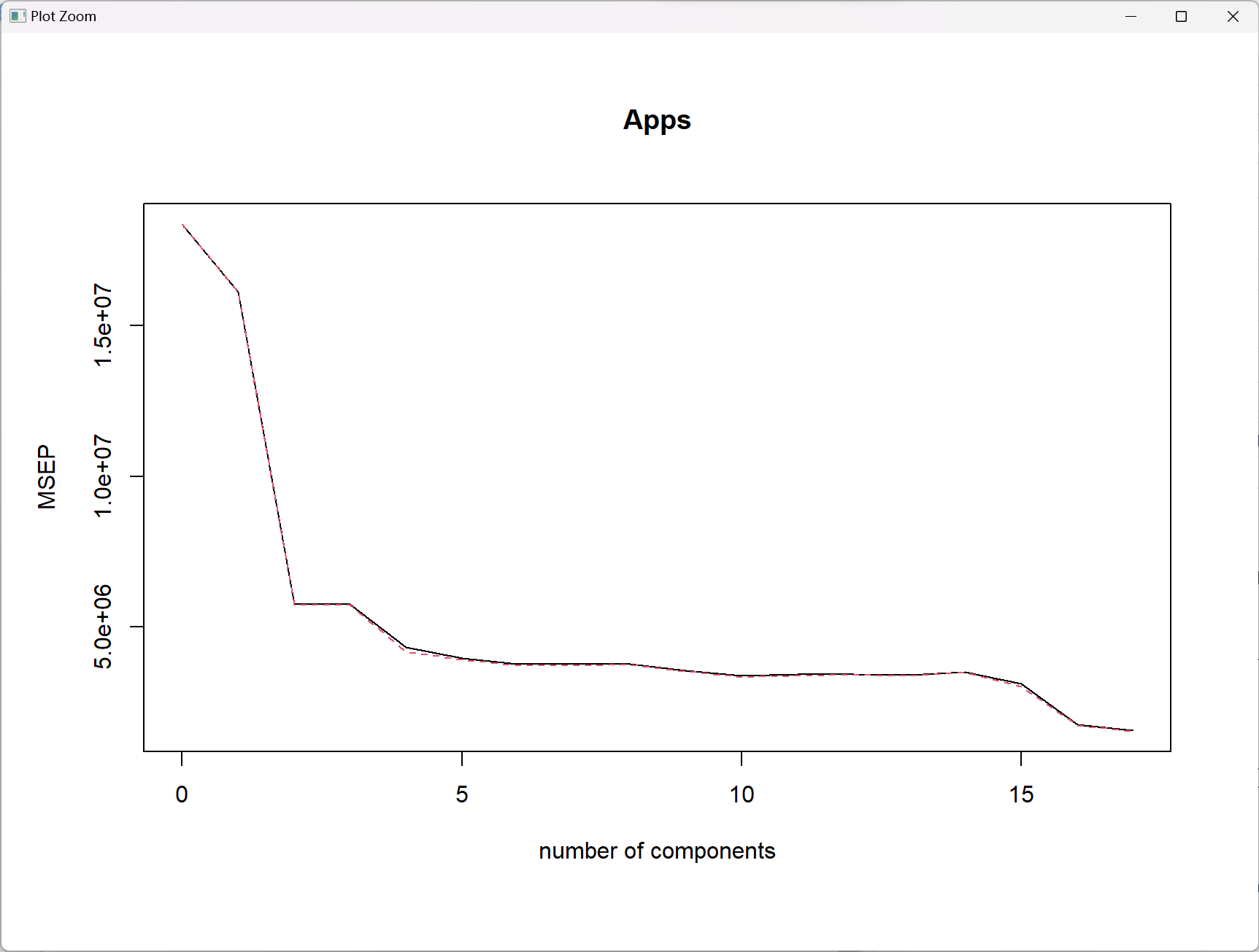
脚本：

library(pls)

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

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

截图：



取M为15。

脚本：

pcr.pred = predict(pcr.fit, College.test, ncomp=15)

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

测试误差：



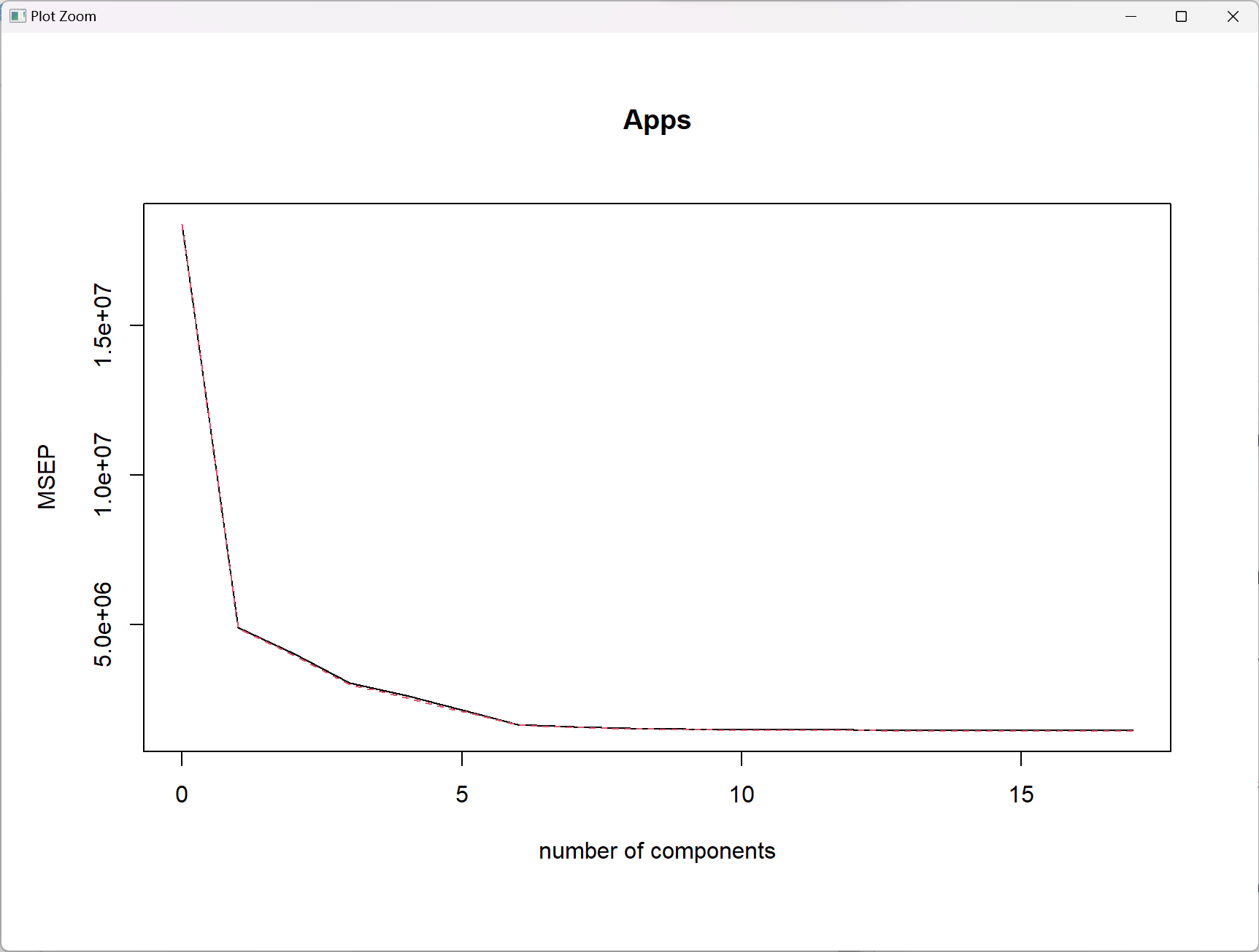
（f）问题（略）

脚本：

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

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

截图：



取M为10

脚本：

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

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

测试误差：

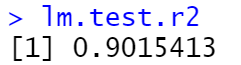


（g）问题（略）

从测试误差的结果来看，最小二乘，岭回归，lasso和PLS模型得到的测试误差处于同一个水平。

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）

脚本：

set.seed(1)

p = 20

n = 1000

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

eps = rnorm(p)

B = rnorm(p)

B[2] = 0

B[4] = 0

B[6] = 0

B[10] = 0

B[16] = 0

y = x %\*% B + eps

（b）问题（略）

脚本：

train = sample(1:1000, size=100, replace = FALSE)

y.train = y[train, ]

y.test = y[-train, ]

x.train = x[train, ]

x.test = x[-train, ]

train=data.frame(y=y.train, x.train)

test=data.frame(y=y.test, x.test)

（c）问题（略）

脚本：

predict.regsubsets = function(object, newdata, id, ...){

form = as.formula(object$call[[2]])

mat = model.matrix(form, newdata)

coefi = coef(object, id=id)

xvars = names(coefi)

mat[,xvars]%\*%coefi

}

library(leaps)

regfit.full = regsubsets(y ~ ., data =train, nvmax = p)

val.errors = rep(NA, p)

for (i in 1:p) {

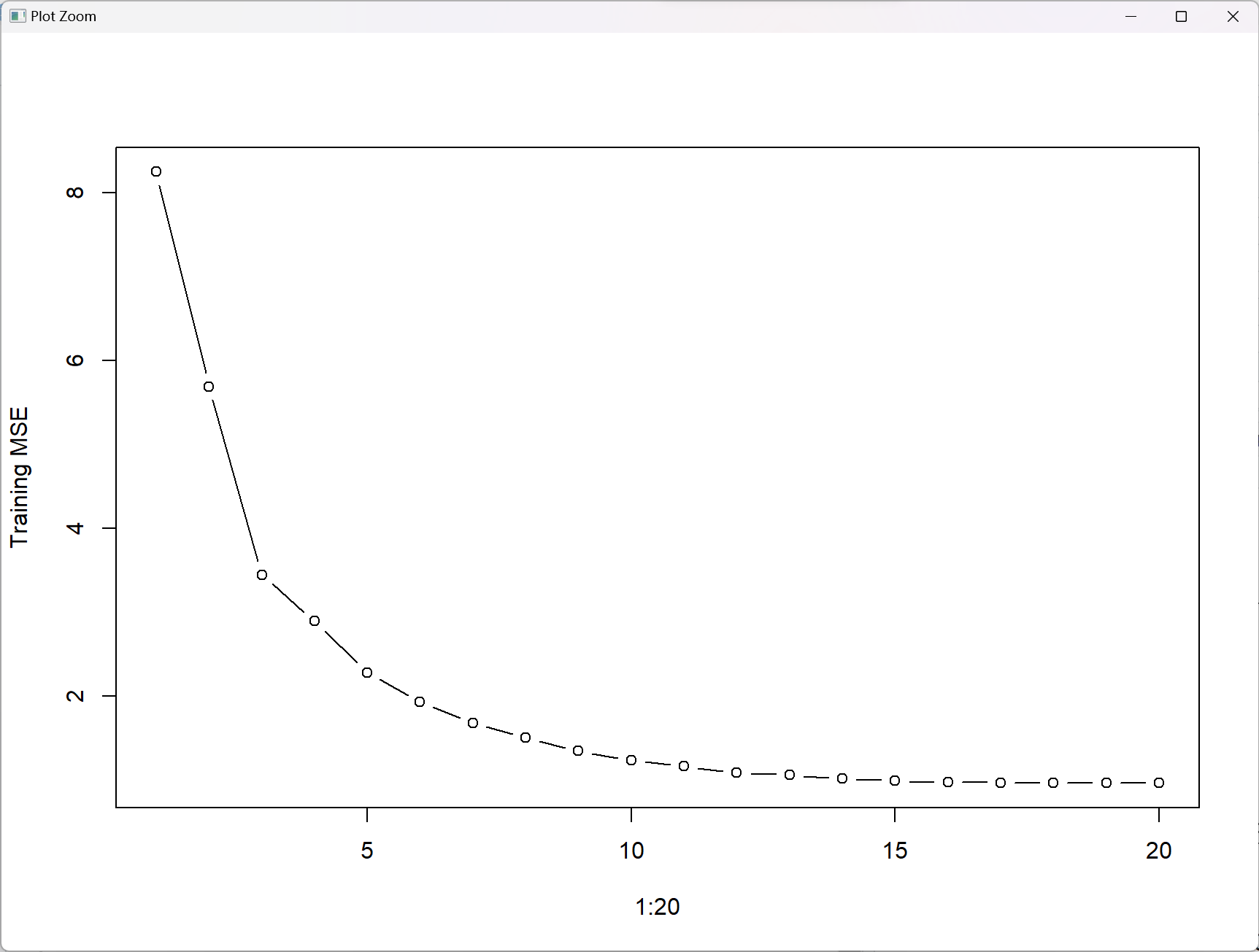
pred = predict(regfit.full,train,id=i)

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

}

plot(1:20, val.errors, ylab = "Training MSE",type="b")

截图：



（d）问题（略）

脚本：

val.errors = rep(NA, p)

for (i in 1:p) {

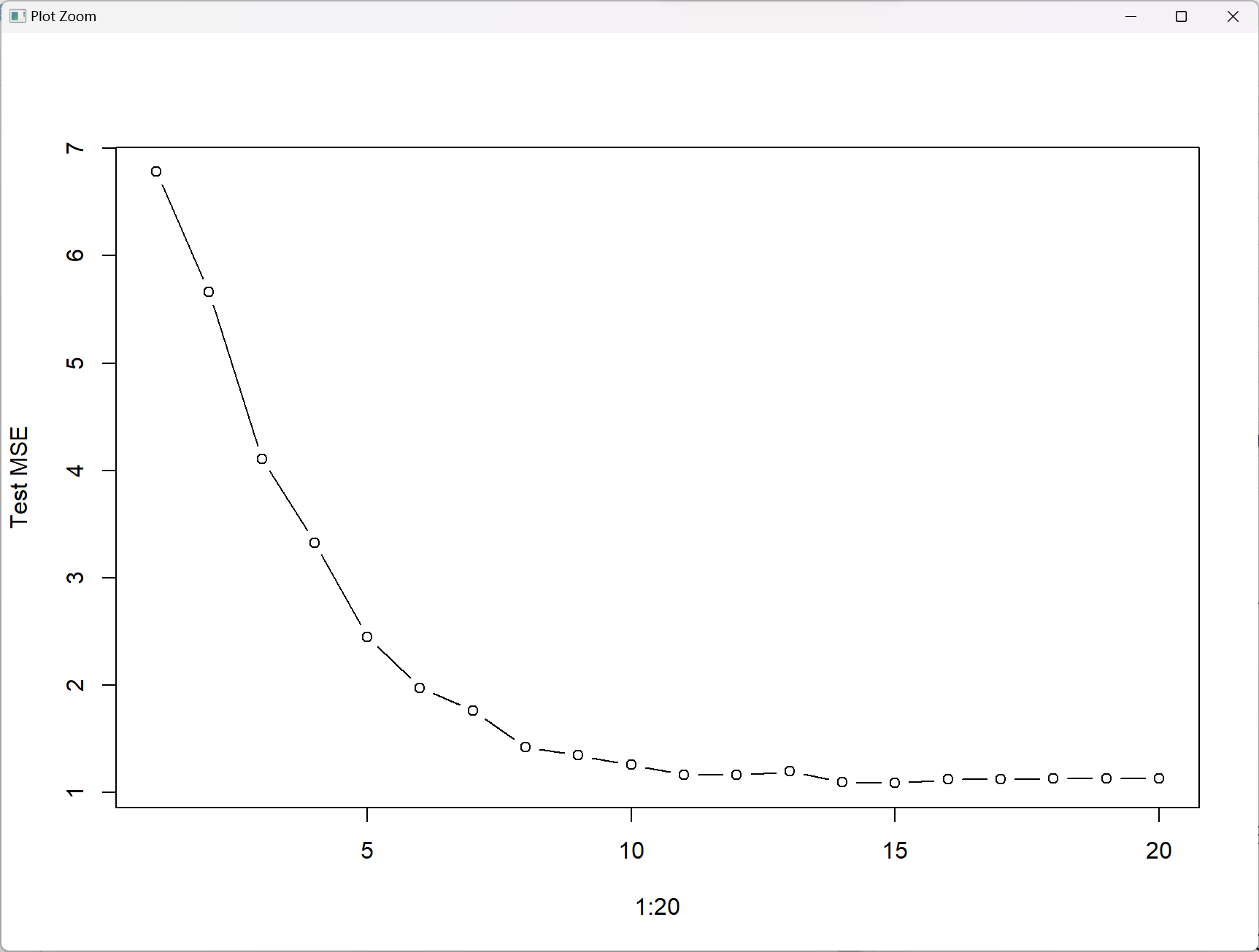
pred = predict(regfit.full,test,id=i)

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

}

plot(1:20,val.errors, ylab = "Test MSE",type = "b")

截图：



（e）问题（略）

脚本：

which.min(val.errors)

截图：



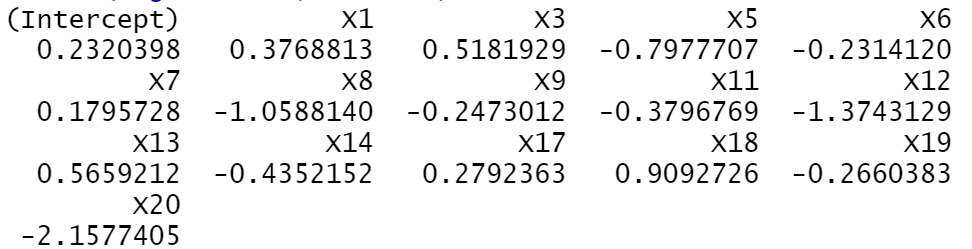
在含有15个特征的时候，测试MSE取值最小。因为在a问中设置了5个b中的元素为0。所以在有15个特征左右的时候，测试MSE应该最小，如上图所示，在特征数为15时，测试MSE最小。

（f）问题（略）

脚本：

coef(regfit.full, id = 15)

截图：



设置的B元素为0中，2，4，10，16成功没有被捕捉到，虽然6捕捉到了并错误地没有捕捉了到15。但与实际模型已经有着较高的拟合度。

（g）问题（略）

脚本：

val.errors = rep(NA, p)

names(B) = paste0("X", 1:20)

for(i in 1:p) {

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

df.err=merge(data.frame(beta=names(B),B),data.frame(beta=names(coef.i),coef.i), all.x=T)

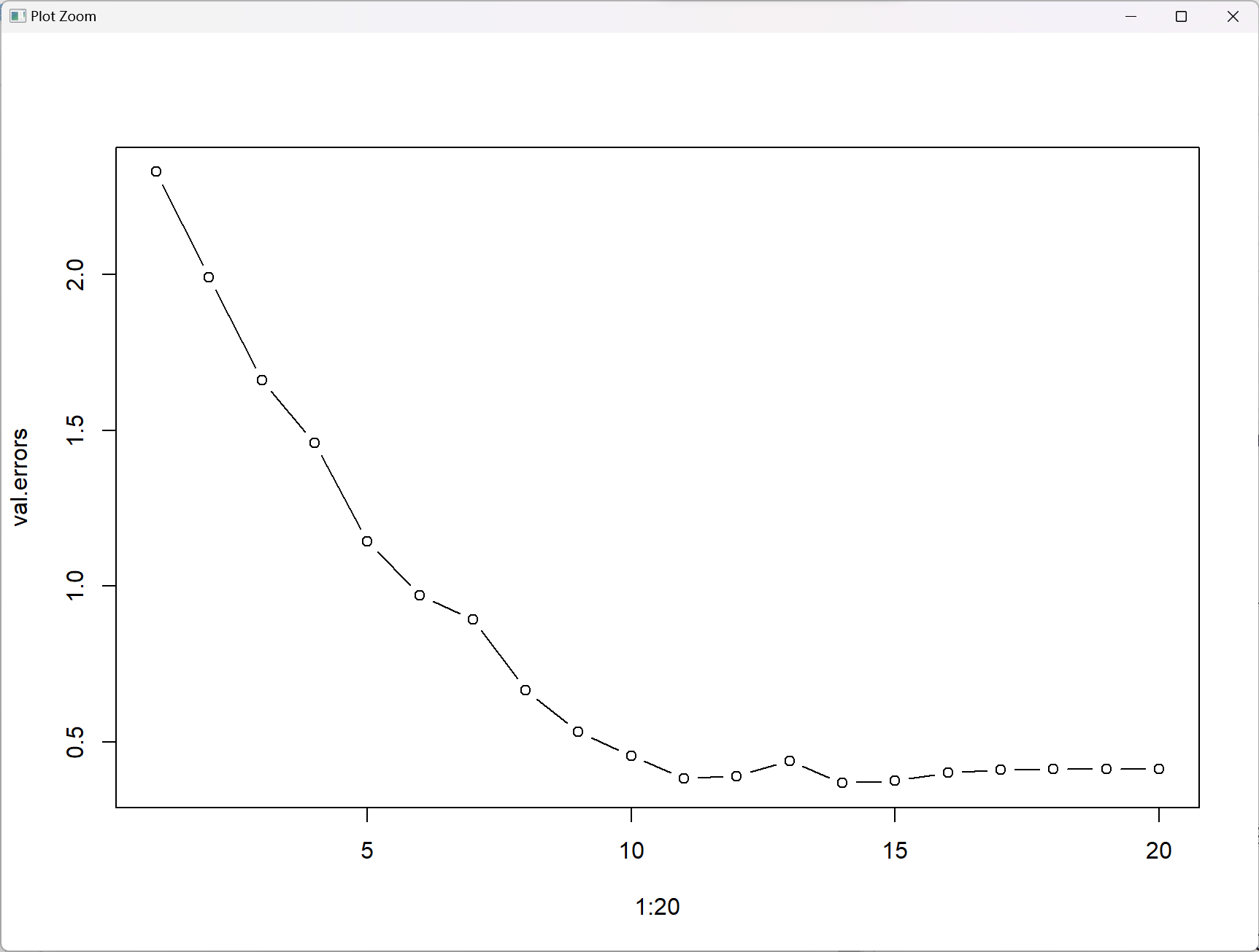
df.err[is.na(df.err[,3]),3] = 0

val.errors [i] = sqrt(sum((df.err[,2] - df.err[,3])^2))

}

plot(1:20, val.errors, type="b",)

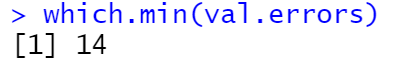
截图：



脚本：

which.min(val.errors)

截图：



与d得到的图像对比，可以看出两个图像十分相似，虽然求出的使error最小的变量个数为14，但从图像中可以看到与15差别不大。