高级统计方法 第7次作业:

序号：1 姓名：杨程瑜 学号：20202241107 班级：网2001

**概念**

3.问题（略）

首先，该式表明其是一个lasso回归模型

1. 问题（略）

iv.稳定减小 正确

训练集上RSS依赖于训练集上的偏差，模型灵活性越高，在训练集上拟合的效果越好，RSS越低。而s增大，相应的λ减小，约束放松。模型更接近最小二乘，模型灵活性增高，故在训练集上拟合效果越好，训练集RSS稳定减小。（当s足够大，直至β落在约束区域内时，不再变化）

1. 问题（略）

ii.最初减小，然后开始增加，图像呈现一个U形 正确

测试集上RSS同时依赖于偏差和方差（MSE均方误差）。前期s增大，相应的λ减小，约束放松。模型更接近最小二乘，模型灵活性增高，偏差减小大于方差增加，均方误差减小。而后期产生过拟合，偏差减小小于方差增大，均方误差增大。

1. 问题（略）

iii.稳定增长 正确

s增大，相应的λ减小，约束放松。模型更接近最小二乘，模型灵活性增高，方差单调增大。（当s足够大，直至β落在约束区域内时，不再变化）

（d）问题（略）

iv.稳定减小 正确

s增大，相应的λ减小，约束放松。模型更接近最小二乘，模型灵活性增高，偏差单调减小。

（e）问题（略）

v.保持不变

不可约误差是拟合模型中固有的不确定性或者称为噪声引入的误差，无论模型的灵活性如何变化，不可约误差都保持不变。所以基本上，不可约误差完全独立于s，故保持不变。

4.问题（略）

首先，该式表明其是一个岭回归模型

（a）问题（略）

iii.稳定增长 正确

训练集上RSS依赖于训练集上的偏差，模型灵活性越高，在训练集上拟合的效果越好，RSS越低。而λ增大，更多预测变量的系数压缩至0，模型更远离最小二乘，模型灵活性降低，故在训练集上拟合效果更差，训练集RSS稳定增长

（b）问题（略）

ii.最初减小，然后开始增加，图像呈现一个U形 正确

测试集上RSS同时依赖于偏差和方差（MSE均方误差）。前期λ增大，约束适度加强。模型更远离最小二乘，模型灵活性降低，过拟合效果得到缓解，偏差增加小于方差减小，均方误差减小。后期约束进一步加强，偏差增大大于方差减小，均方误差增大。

（c）问题（略）

iv.稳定减小 正确

λ增大，约束加强，β向0压缩。模型更远离最小二乘，模型灵活性下降，方差单调减小。

（d）问题（略）

iii.稳定增加 正确

λ增大，约束加强，β向0压缩。模型更远离最小二乘，模型灵活性下降，偏差单调增大。

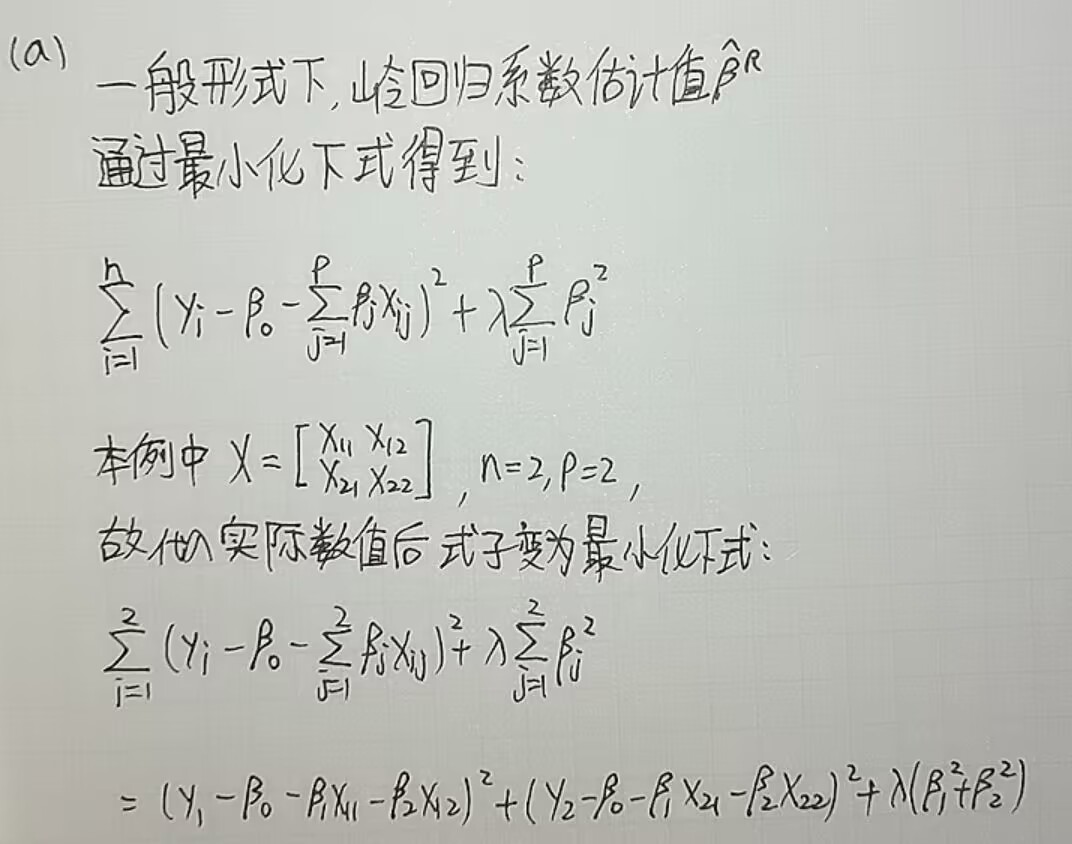
（e）问题（略）

v.保持不变

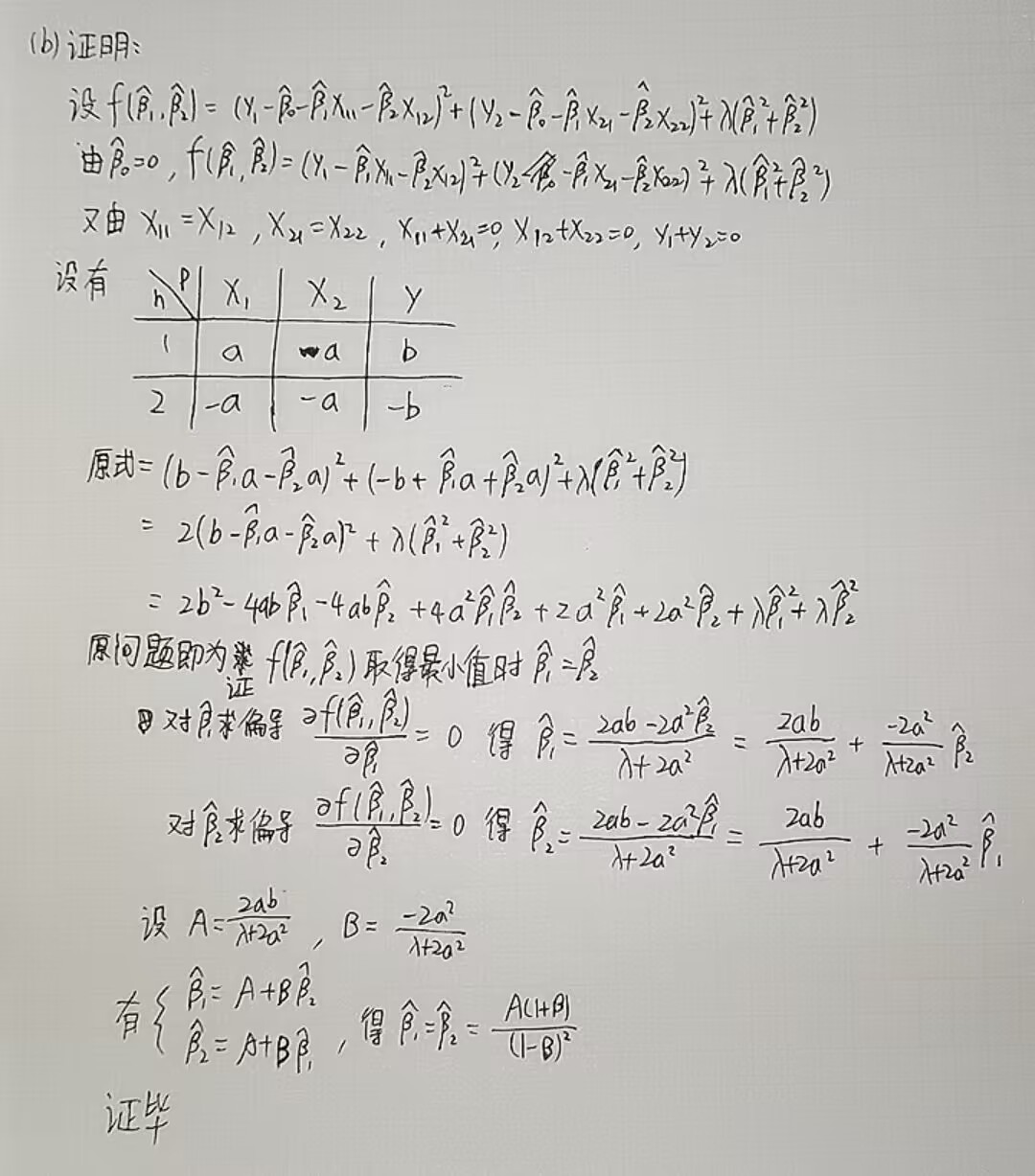
不可约误差是拟合模型中固有的不确定性或者称为噪声引入的误差，无论模型的灵活性如何变化，不可约误差都保持不变。所以基本上，不可约误差完全独立于λ，故保持不变。

5.问题（略）

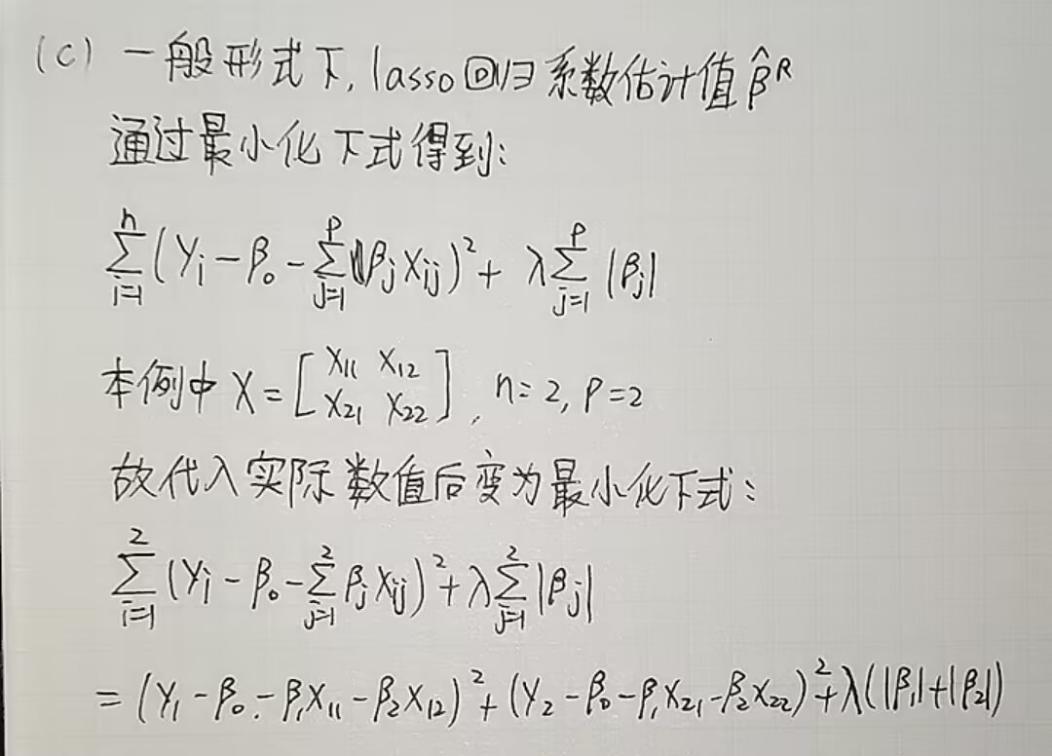
（a）问题（略）



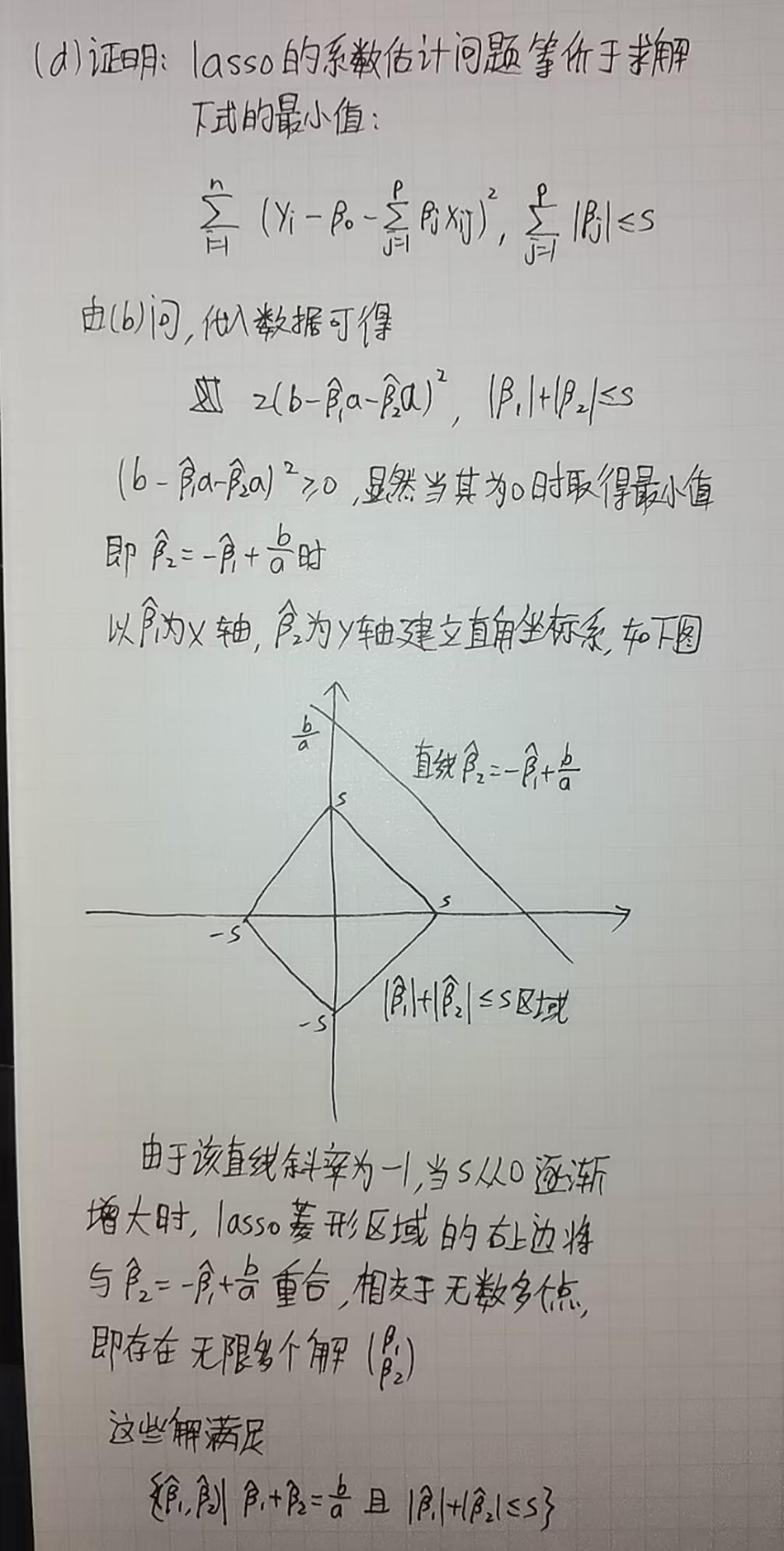
（b）问题（略）



（c）问题（略）

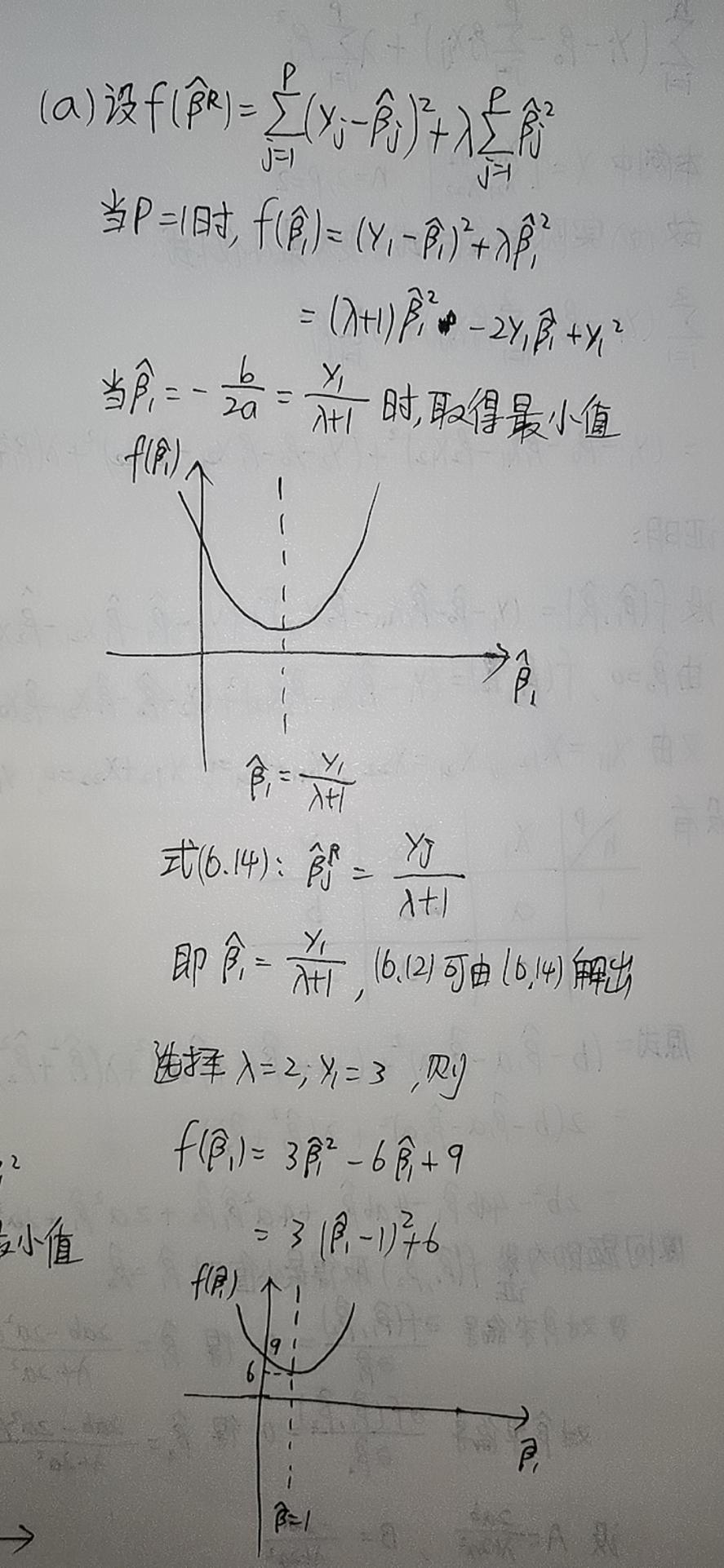


（d）问题（略）

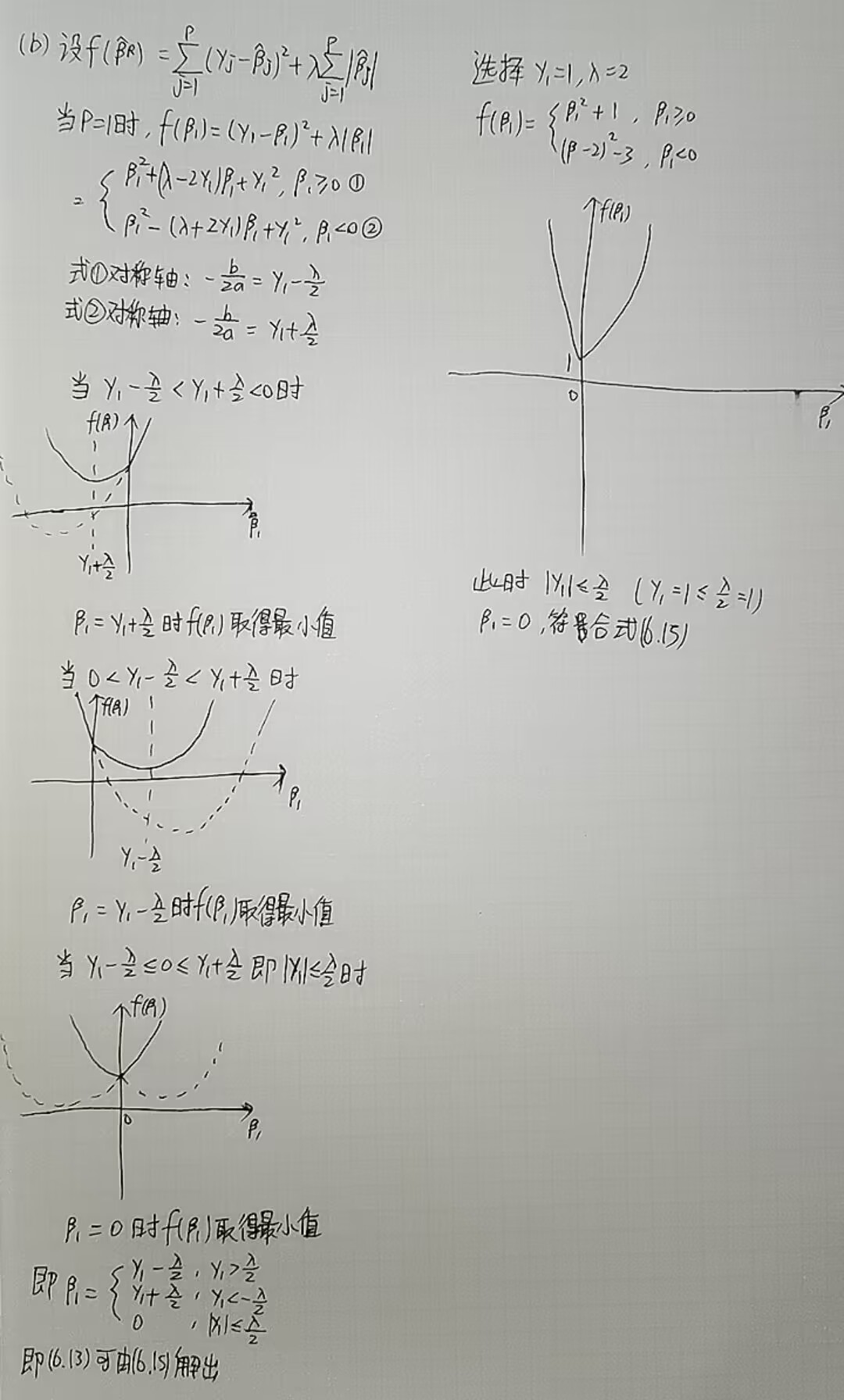


6.问题（略）

（a）问题（略）

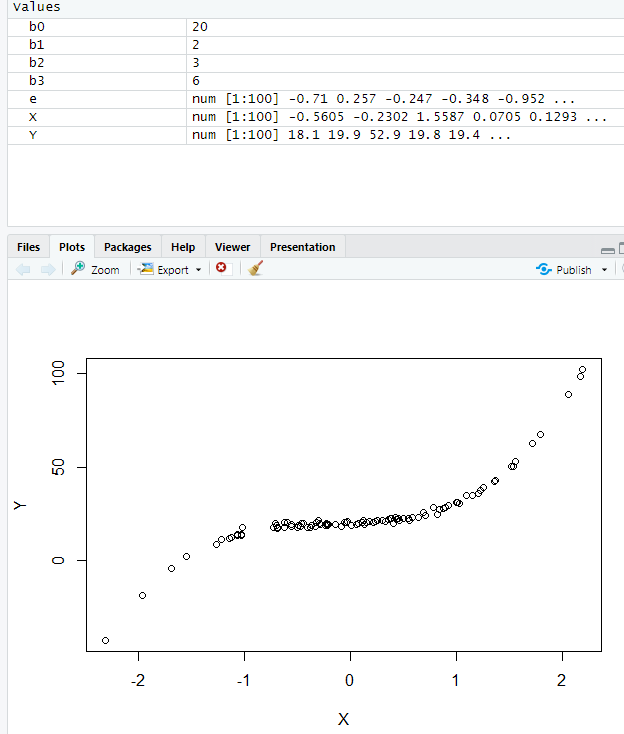
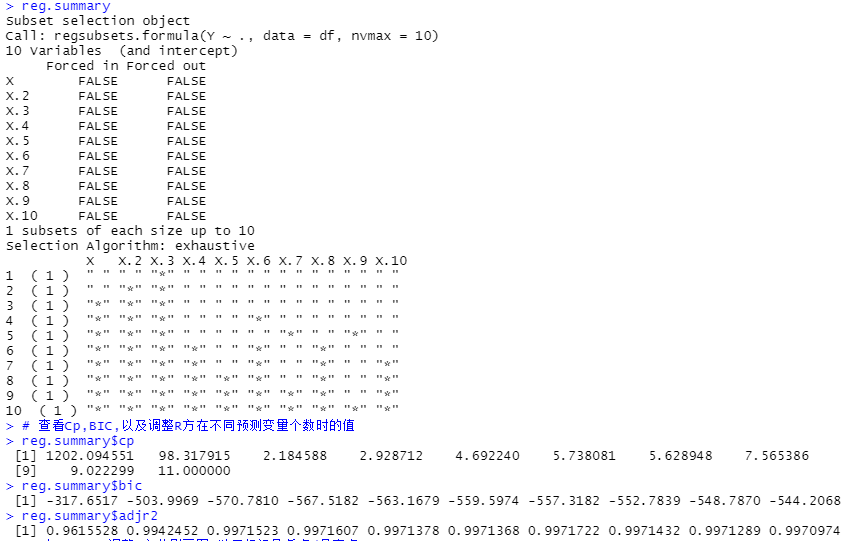


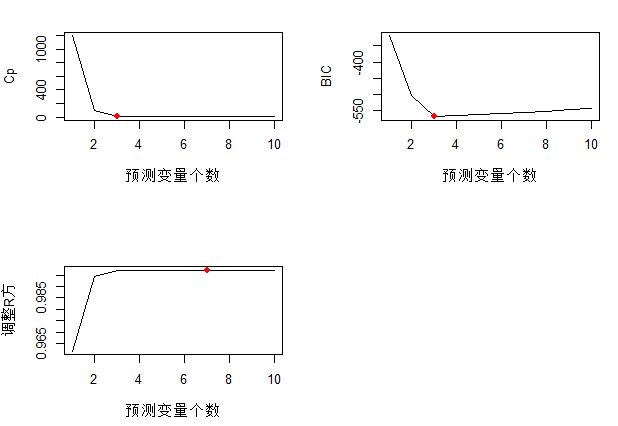
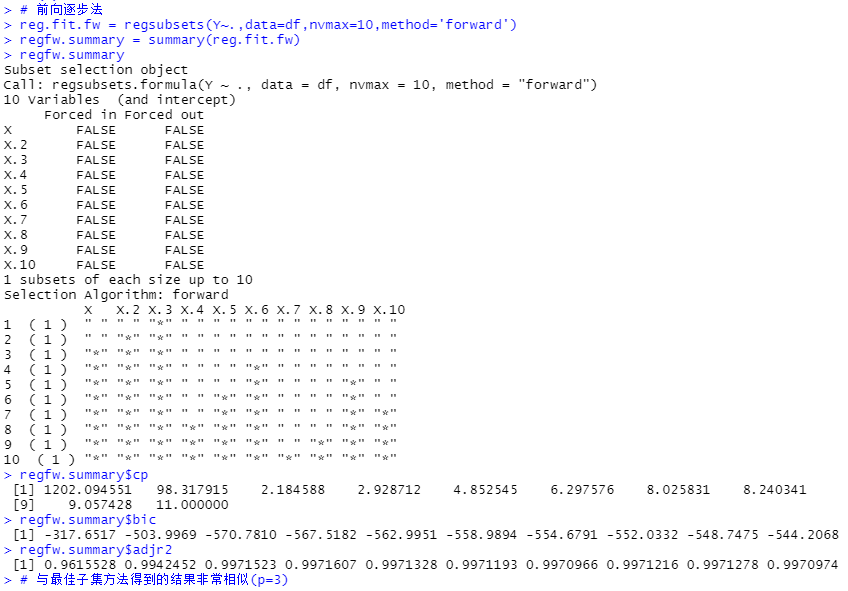
（b）问题（略）

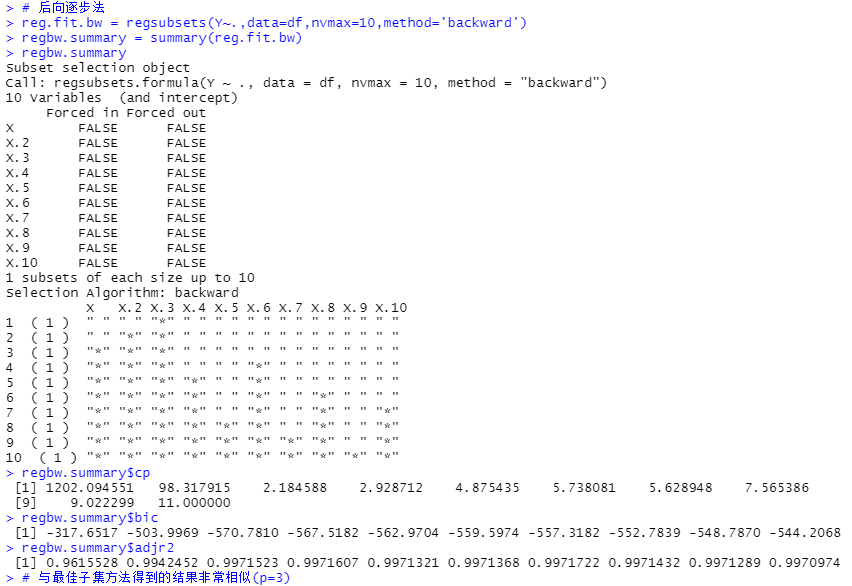
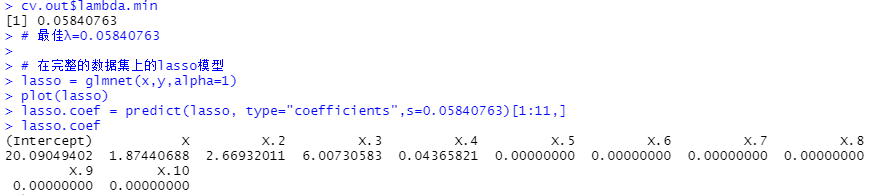


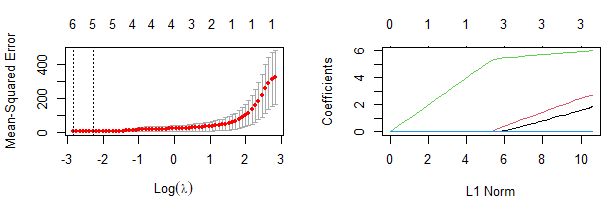
**应用**

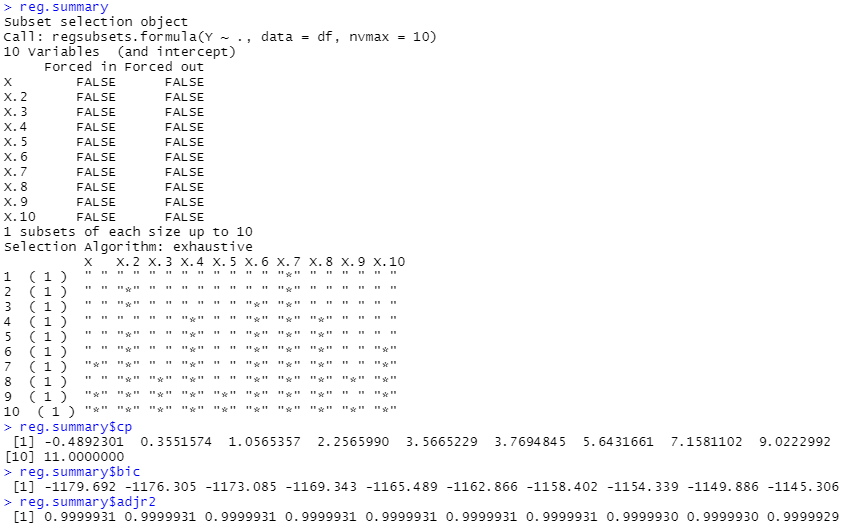
8.问题（略）

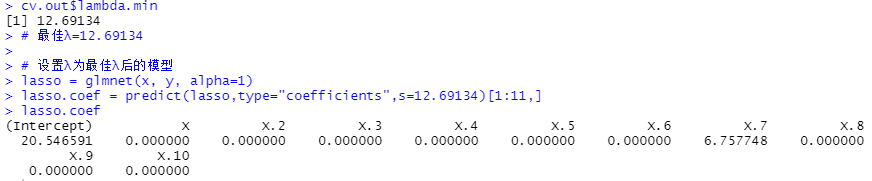
# 6.8.8
  
rm(list=ls())
  
（a）问题（略）  
set.seed(123)
  
X = rnorm(100)
  
e = rnorm(100, mean=0, sd=1)
  
  
（b）问题（略）  
b0 = 20
  
b1 = 2
  
b2 = 3
  
b3 = 6
  
Y = b0 + b1\*X + b2\*X^2 + b3\*X^3 + e
  
plot(X ,Y)
  
  
（c）问题（略）  
# 引入leaps库，其中包含regsubsets函数
  
library(leaps)
  
# 构造dataframe(X~X^10以及Y)
  
df = data.frame(X,X^2,X^3,X^4,X^5,X^6,X^7,X^8,X^9,X^10,Y)
  
# 选择Y的所有预测变量，最大变量个数为10
  
reg.fit = regsubsets(Y~.,data=df,nvmax=10)
  
reg.summary = summary(reg.fit)
  
# 通过summary可以看到每一步选择了哪些变量
  
reg.summary
  
# 查看Cp,BIC,以及调整R方在不同预测变量个数时的值
  
reg.summary$cp
  
reg.summary$bic
  
reg.summary$adjr2
  
# 对Cp,BIC,调整R方分别画图,以及标记最低点/最高点
  
par(mfrow=c(2,2))
  
plot(reg.summary$cp,xlab="预测变量个数", ylab="Cp",type="l")
  
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col = "red", pch = 16)
  
plot(reg.summary$bic,xlab="预测变量个数", ylab="BIC",type="l")
  
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col = "red", pch = 16)
  
plot(reg.summary$adjr2,xlab="预测变量个数", ylab="调整R方",type="l")
  
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col = "red", pch = 16)
  
  
# 从图中可以看出Cp下的最优模型为p=3，BIC下为p=3，调整R方下为p=7
  
# 但是调整R方下当变量个数达到三个后，其值上升变化不大
  
# 故总体来说，当p=3时的模型是最优模型
  


  
（d）问题（略）  
# 前向逐步法
  
reg.fit.fw = regsubsets(Y~.,data=df,nvmax=10,method='forward')
  
regfw.summary = summary(reg.fit.fw)
  
regfw.summary
  
regfw.summary$cp
  
regfw.summary$bic
  
regfw.summary$adjr2
  
# 与最佳子集方法得到的结果非常相似(p=3)
  
  
# 后向逐步法
  
reg.fit.bw = regsubsets(Y~.,data=df,nvmax=10,method='backward')
  
regbw.summary = summary(reg.fit.bw)
  
regbw.summary
  
regbw.summary$cp
  
regbw.summary$bic
  
regbw.summary$adjr2
  
# 与最佳子集方法得到的结果非常相似(p=3)
  


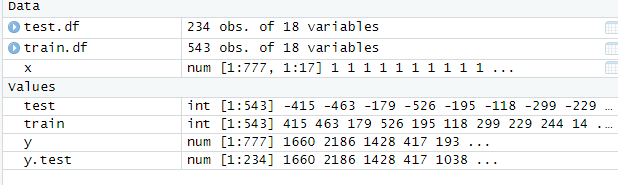
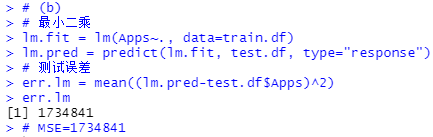
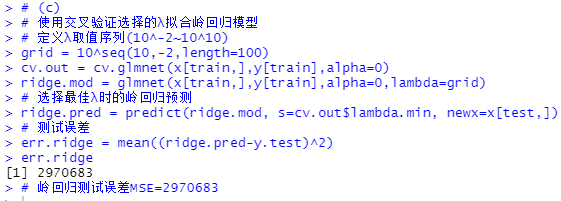
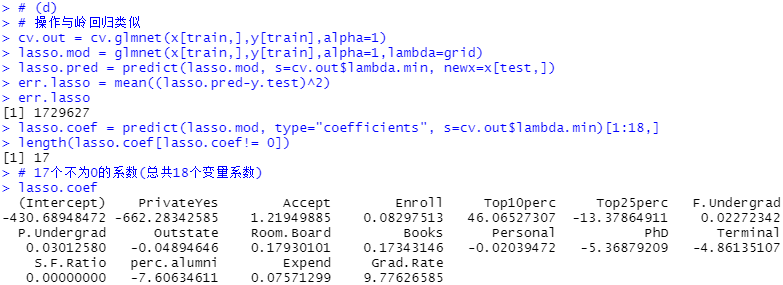
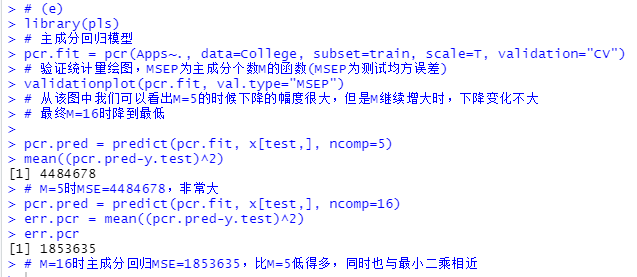
  
（e）问题（略）  
library(glmnet)
  
set.seed(123)
  
# 从dataframe中构造X矩阵和Y向量
  
x = model.matrix(Y~.,df)[,-1]
  
y = df$Y
  
# 划分训练集和测试集(随机1:1)
  
train = sample(1:nrow(x), nrow(x)/2)
  
test = (-train)
  
y.test = y[test]
  
# 训练集下的lasso模型
  
lasso.train = glmnet(x[train,], y[train], alpha=1)
  
plot(lasso.train)
  
# 交叉验证误差与λ的函数与图像
  
cv.out = cv.glmnet(x[train,],y[train], alpha=1)
  
plot(cv.out)
  
cv.out$lambda.min
  
# 最佳λ=0.05840763
  
  
# 在完整的数据集上的lasso模型
  
lasso = glmnet(x,y,alpha=1)
  
plot(lasso)
  
lasso.coef = predict(lasso, type="coefficients",s=0.05840763)[1:11,]
  
lasso.coef
  
  
# 从图像以及参数的估计能够看出此模型在lasso下p=4时为最优模型
  
# 结论是该模型是一个有四个预测变量(X,X^2,X^3,X^4)的稀疏模型
  
# 同时预估结果与我们给定的系数非常相近(虽然没有给定b4,但是b4预估结果接近0)
  
# b0 = 20 b1 = 2 b2 = 3 b3 = 6 b4 = 0
  
# 估计值
  
# b0 = 20.09049402 b1 = 1.87440688 b2 = 2.66932011 b3 = 6.00730583 b4 = 0.04365821


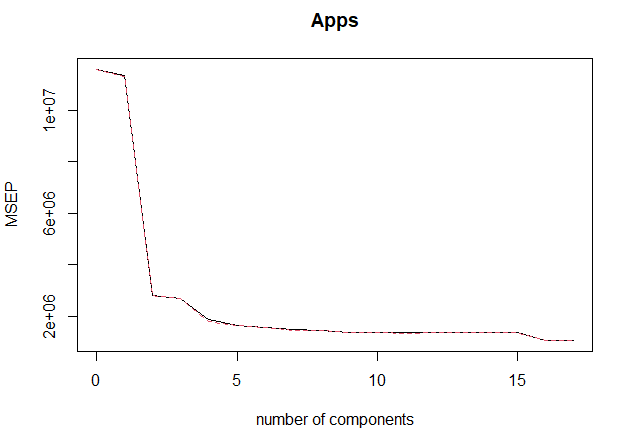
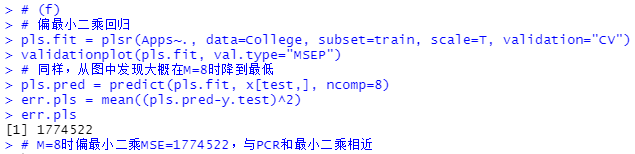
  
  
（f）问题（略）  
set.seed(123)
  
# 定义函数和dataframe
  
b7 = 7
  
Y = b0 + b7\*X^7 + e
  
df = data.frame(X,X^2,X^3,X^4,X^5,X^6,X^7,X^8,X^9,X^10,Y)
  
  
# 最佳子集选择
  
reg.fit = regsubsets(Y~.,data=df,nvmax=10)
  
reg.summary = summary(reg.fit)
  
reg.summary
  
reg.summary$cp
  
reg.summary$bic
  
reg.summary$adjr2
  
# 整体来说p=3时达到最优模型
  
  
# lasso模型
  
# 训练集与数据集
  
x = model.matrix(Y~.,df)[,-1]
  
y = df$Y
  
train = sample(1:nrow(x), nrow(x)/2)
  
test = (-train)
  
y.test = y[test]
  
  
# 交叉验证误差与λ的函数
  
cv.out = cv.glmnet(x[train,],y[train], alpha=1)
  
cv.out$lambda.min
  
# 最佳λ=12.69134
  
  
# 设置λ为最佳λ后的模型
  
lasso = glmnet(x, y, alpha=1)
  
lasso.coef = predict(lasso,type="coefficients",s=12.69134)[1:11,]
  
lasso.coef
  
  
# b7的估计值6.757748 和 真实值7非常接近

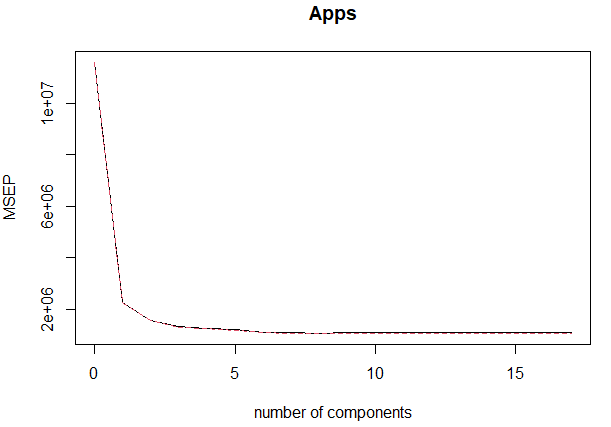


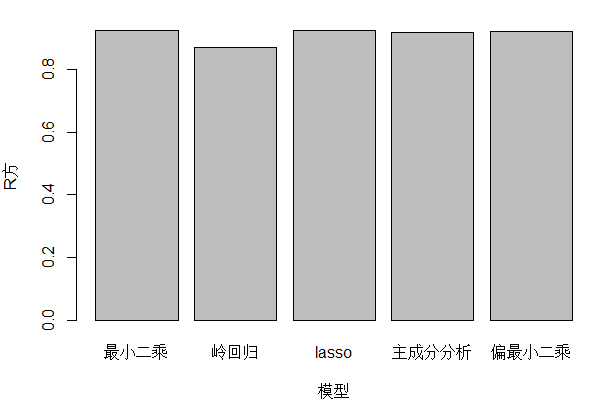


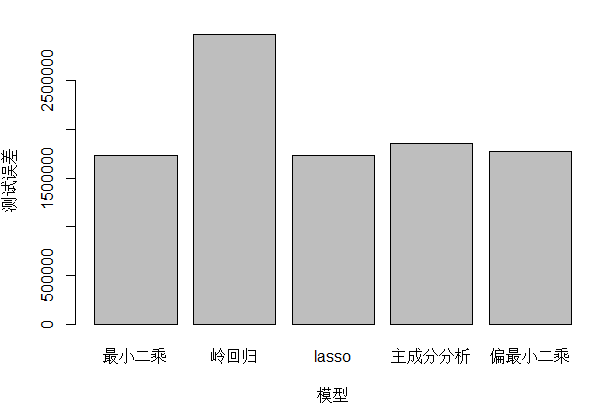
9.问题（略）

# 6.8.9
  
rm(list=ls())
  
library(ISLR)
  
library(glmnet)
  
attach(College)
  
（a）问题（略）  
set.seed(123)
  
# x为其他变量,y为申请人数
  
x = model.matrix(Apps~.,College)[,-1]
  
y = College$Apps
  
# 划分训练集与测试集(随机7:3)
  
train = sample(1:nrow(x), nrow(x)\*0.7)
  
test = (-train)
  
y.test = y[test]
  
train.df = data.frame(College[train,])
  
test.df = data.frame(College[test,])
  
  
（b）问题（略）  
# 最小二乘
  
lm.fit = lm(Apps~., data=train.df)
  
lm.pred = predict(lm.fit, test.df, type="response")
  
# 测试误差
  
err.lm = mean((lm.pred-test.df$Apps)^2)
  
err.lm
  
# MSE=1734841
  
  
（c）问题（略）  
# 使用交叉验证选择的λ拟合岭回归模型
  
# 定义λ取值序列(10^-2~10^10)
  
grid = 10^seq(10,-2,length=100)
  
cv.out = cv.glmnet(x[train,],y[train],alpha=0)
  
ridge.mod = glmnet(x[train,],y[train],alpha=0,lambda=grid)
  
# 选择最佳λ时的岭回归预测
  
ridge.pred = predict(ridge.mod, s=cv.out$lambda.min, newx=x[test,])
  
# 测试误差
  
err.ridge = mean((ridge.pred-y.test)^2)
  
err.ridge
  
# 岭回归测试误差MSE=2970683
  
  
（d）问题（略）  
# 操作与岭回归类似
  
cv.out = cv.glmnet(x[train,],y[train],alpha=1)
  
lasso.mod = glmnet(x[train,],y[train],alpha=1,lambda=grid)
  
lasso.pred = predict(lasso.mod, s=cv.out$lambda.min, newx=x[test,])
  
err.lasso = mean((lasso.pred-y.test)^2)
  
err.lasso
  
lasso.coef = predict(lasso.mod, type="coefficients", s=cv.out$lambda.min)[1:18,]
  
length(lasso.coef[lasso.coef!= 0])
  
# 17个不为0的系数(总共18个变量系数)
  
lasso.coef
  
  
# lasso回归测试误差MSE=1729378，略低于最小二乘的MSE
  
# 只有S.F.Ratio变量的系数为0，但是很多变量的系数几乎为0
  
  
（e）问题（略）  
library(pls)
  
# 主成分回归模型
  
pcr.fit = pcr(Apps~., data=College, subset=train, scale=T, validation="CV")
  
# 验证统计量绘图，MSEP为主成分个数M的函数(MSEP为测试均方误差)
  
validationplot(pcr.fit, val.type="MSEP")
  
# 从该图中我们可以看出M=5的时候下降的幅度很大，但是M继续增大时，下降变化不大
  
# 最终M=16时降到最低
  
  
pcr.pred = predict(pcr.fit, x[test,], ncomp=5)
  
mean((pcr.pred-y.test)^2)
  
# M=5时MSE=4484678，非常大
  
pcr.pred = predict(pcr.fit, x[test,], ncomp=16)
  
err.pcr = mean((pcr.pred-y.test)^2)
  
err.pcr
  
# M=16时主成分回归MSE=1853635，比M=5低得多，同时也与最小二乘相近
  


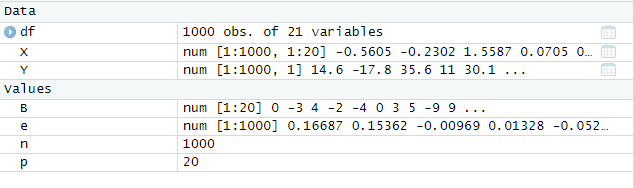
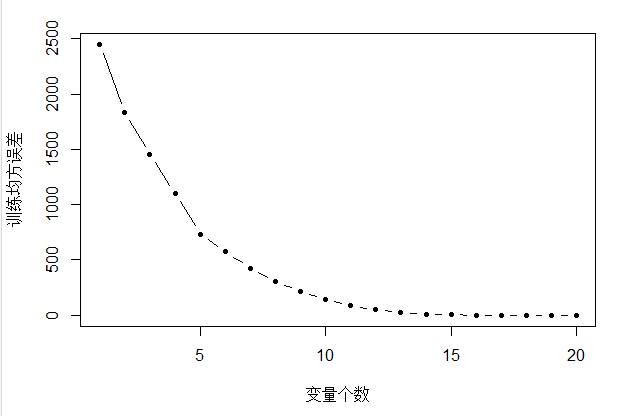
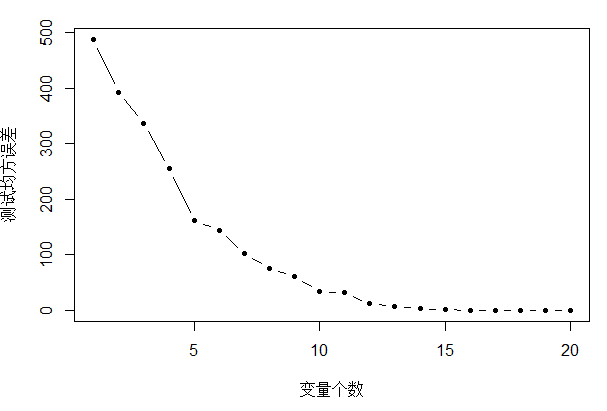
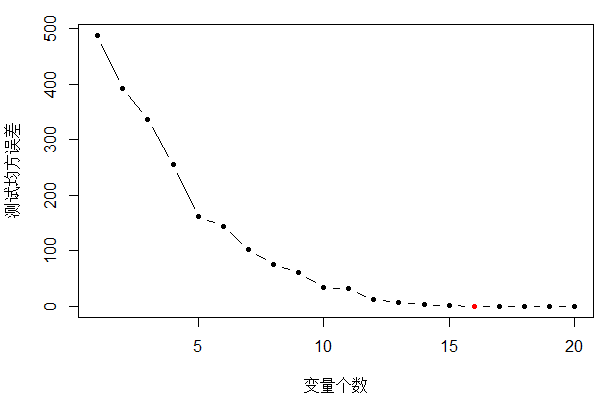
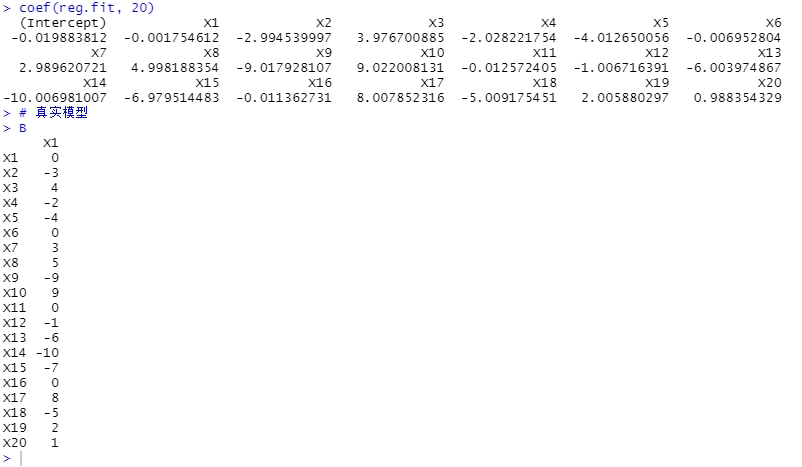
  
（f）问题（略）  
# 偏最小二乘回归
  
pls.fit = plsr(Apps~., data=College, subset=train, scale=T, validation="CV")
  
validationplot(pls.fit, val.type="MSEP")
  
# 同样，从图中发现大概在M=8时降到最低
  
pls.pred = predict(pls.fit, x[test,], ncomp=8)
  
err.pls = mean((pls.pred-y.test)^2)
  
err.pls
  
# M=8时偏最小二乘MSE=1774522，与PCR和最小二乘相近
  


  
（g）问题（略）  
# 计算预测变量测试值平均值(y bar)
  
test.avg = mean(y.test)
  
# 计算各个模型的R方(1-RSS/TSS)
  
lm.r2 = 1 - mean((lm.pred - y.test)^2) / mean((test.avg - y.test)^2)
  
ridge.r2 = 1 - mean((ridge.pred - y.test)^2) / mean((test.avg - y.test)^2)
  
lasso.r2 = 1 - mean((lasso.pred - y.test)^2) / mean((test.avg - y.test)^2)
  
pcr.r2 = 1 - mean((pcr.pred - y.test)^2) / mean((test.avg - y.test)^2)
  
pls.r2 = 1 - mean((pls.pred - y.test)^2) / mean((test.avg - y.test)^2)
  
barplot(c(lm.r2, ridge.r2, lasso.r2, pcr.r2, pls.r2), xlab="模型", ylab="R方",names=c("最小二乘", "岭回归", "lasso", "主成分分析", "偏最小二乘"))
  
# 从柱状图能够看出所有模型的R方都在0.9附近，岭回归模型R方低一些
  
  
err.all = c(err.lm, err.ridge, err.lasso, err.pcr, err.pls)
  
barplot(err.all, xlab="模型", ylab="测试误差", names=c("最小二乘", "岭回归", "lasso", "主成分分析", "偏最小二乘"))
  
# 发现岭回归模型的误差明显别的更大，其他四个模型MSE几乎相同





10.问题（略）

# 6.8.10
  
library(leaps)
  
rm(list=ls())
  
（a）问题（略）  
set.seed(123)
  
n = 1000
  
p = 20
  
# n\*p形正态分布随机数据矩阵X
  
X = matrix(rnorm(n\*p), n, p)
  
# 设定系数矩阵B(-10到10范围随机选20个)
  
B = sample(-10:10, 20)
  
# 随机让一些系数为0
  
B[1] = 0
  
B[6] = 0
  
B[11] = 0
  
B[16] = 0
  
e = rnorm(1000, mean=0, sd=0.1)
  
# %\*%为矩阵乘法(X[n,p] %\*% B[p,1])
  
Y = X%\*%B + e
  
df = data.frame(X, Y)
  
  
（b）问题（略）  
# 训练集与测试集(随机9:1)
  
sample = sample.split(df$Y, 0.10)
  
train = subset(df, sample==T)
  
test = subset(df, sample==F)
  
  
（c）问题（略）  
# 最佳子集选择
  
reg.fit = regsubsets(Y~., data=train, nvmax=20)
  
reg.summary = summary(reg.fit)
  
# 不同变量个数的训练集MSE
  
train.MSE = (reg.summary$rss)/length(train)
  
plot(1:20,train.MSE,xlab = "变量个数",ylab = "训练均方误差", pch = 20, type = "b")
  
  
# 训练均方误差确实随选择的变量个数的增加而单调下降
  
  
（d）问题（略）  
library(HH)
  
test.MSE = rep(NA,20)
  
# 遍历i个变量模型
  
for(i in 1:20){
  
 model=lm.regsubsets(reg.fit, i)
  
 model.pred = predict(model, newdata=test, type="response")
  
 # 计算该模型的MSE
  
 test.MSE[i] = mean((test$Y-model.pred)^2)
  
}
  
# Plot
  
plot(1:20,test.MSE,xlab = "变量个数",ylab = "测试均方误差", pch = 20, type = "b")
  
  
# 测试均方误差也随选择的变量个数的增加而下降，但是曲线没那么平滑
  
  
（e）问题（略）  
# 寻找曲线的最低点并标红
  
points(which.min(test.MSE), test.MSE[which.min(test.MSE)], col = "red", pch = 20)
  
# 发现当变量个数为16时，测试均方误差达到最小值
  
# 模型变量个数增加，灵活性也增加，它能够更好地拟合数据集，导致测试 MSE 迅速下降，直到达到最小值
  
# 然后模型灵活性的进一步增加会导致过拟合，从而导致测试 MSE 的增加
  
  
（f）问题（略）  
# 测试集模型
  
coef(reg.fit, 20)
  
# 真实模型
  
B
  
  
# 逐个对比发现测试集模型与真实模型的系数差距很小，几乎相同
  
  
（g）问题（略）  
# Original coefficient values transposed and columns renamed.
  
B = as.data.frame(t(B))
  
# 将B的结构改为与coef返回值一致(增加名字Xi),方便dataframe直接相减
  
names(B) = paste0('X', 1:(ncol(B)))
  
  
# 循环计算
  
coef.err = rep(NA,20)
  
for (i in 1:20){
  
 a = coef(reg.fit, i)
  
 # B[names(a)[-1]]为真实系数向量
  
 # a[-1]为估计系数向量，即coef(reg.fit, i)[-1]
  
 # [-1]即去除截距项
  
 coef.err[i] = sqrt(sum(((B[names(a)[-1]]-a[-1])^2)))
  
}
  
# 与(d)中test.MSE对比
  
par(mfrow=c(1,2))
  
plot(1:20,test.MSE,xlab = "变量个数",ylab = "测试均方误差", main="(d)", pch = 20, type = "b")
  
plot(1:20,coef.err,xlab = "变量个数",ylab = "系数误差", main="(g)", pch = 20, type = "b")
  
# 标记最小误差
  
points(which.min(coef.err), coef.err[which.min(coef.err)], col = "red", pch = 16)
  
  
# 根据对比图像，我们发现共同的地方是都在变量个数等于16处达到最小误差
  
# 但是变量个数比较少时，(g)中系数误差的波动比较大，并不是单调下降
  
# 某种程度上通过给出有较小系数误差的模型不一定能有较低的测试均方误差(比如第一个点)

