**### 设定即将使用的文件目录，这样读取数据就可以直接输入文件名**

setwd("C:/STONY/Practice/R (No.8)")

**###** **读取文件**

dat <- read.table("data.csv", header=TRUE, sep=",")

**###** **将数据里的缺失值去除，你们不需要做这一步，老师给的数据不需要再做任何处理**

dat <- dat[complete.cases(dat),] # remove rows with missing value

**###** **str() 是查看数据信息，例如有多少个obs,多少个变量**

str(dat) # 1364 obs. of 32 variables

**###** **查看Y与其他变量之间的相关性**

cor(dat$Y,dat[,])

**###** **去除Y值，查看变量间的相关性**

cor(dat[,-1])

which(cor(dat[,-1])!= 1 & abs(cor(dat[,-1]))>0.1) # return 'integer(0)'

##########################################################################

**###** **建立线性回归模型，. 代表包含所有的一次项**

fit1 <- lm(Y~.,data=dat)

**###** **查看建立的模型的基本信息，\*越多代表变量越显著**

summary(fit1)

**###** **如果**pvalue < 0.0**1 则证明变量显著，** # E1, E6, G9 代表得到的显著的结果

pvalue1 <-NULL

pvalue1 <-c(summary(fit1)$coefficient[,4])

pvalue1[which(pvalue1<0.01)]

# E1, E6, G9 significant

**###** **建立新的模型** .^2 **代表包含所有的一次项和二次项，一次类推** (eg, E1:E6)

fit2 <- lm(Y~.^2,data=dat)

summary(fit2)

pvalue2 <-NULL

pvalue2 <-c(summary(fit2)$coefficient[,4])

pvalue2[which(pvalue2<0.01)]

# E3, E1:E6, E3:G17, E6:G16 significant

##########################################################################

**###筛选出以上结果的显著变量，数字为几代表第几列**

dat1 <- dat[c(1,2,4,7,16,23,24)]

str(dat1) # Y, E1, E3, E6, G9, G16, G17

### stepwise regression ###

**### null 表示没有变量，full表示包含一次，二次，三次，四次项的模型**

null <- lm(Y~1,data=dat1)

full <- lm(Y~.^4,data=dat1)

**###利用逐步回归法 step() 筛选出以上变量中更显著的变量，这里是从full到null的逐步**

step(full,scope=list(upper=null),data=dat1,direction='both',k=log(1364))

# Y ~ E1 + E6 + G9 + G16 + E1:E6 + E6:G16

**### 得到以上变量后又继续建立模型，继续筛选**

fit3 <- lm(Y ~ E1 + E6 + G9 + G16 + E1:E6 + E6:G16,data=dat1)

**###查看模型的结果**

summary(fit3)

# G9, E1:E6 significant

# Adjusted R-squared = 0.4532

**###查看该模型的BIC,越小越好**

BIC(fit3) # 73050.24

**###利用以上筛选出的变量再建模，得到最终从full 到null回归的最终结果**

fit4 <- lm(Y ~ G9 + E1:E6 -1,data=dat1)

summary(fit4)

# Adjusted R-squared = 0.6765

BIC(fit4) # 73027.52

### stepwise regression ###

**###除了从full 到null ,还需要从null 到full逐步回归，进行第二次大筛选**

step(null,scope=list(upper=full),data=dat1,direction='both',k=log(1364))

# Y ~ E6 + E1 + G9 + E6:E1

**### 得到显著变量后，建模，查看BIC**

fit5 <- lm(Y ~ E6 + E1 + G9 + E6:E1,data=dat1)

summary(fit5)

# G9, E6:E1 signiificant

# Adjusted R-squared = 0.4509

BIC(fit5) # 73043.32

**###又用以上的筛选变量，再建模，查看BIC**

fit6 <- lm(Y ~ G9 + E6:E1 -1,data=dat1)

summary(fit6)

# Adjusted R-squared = 0.6765

BIC(fit6) # 73027.52

**### lasso ###**

**除了逐步回归的方法，我们还可以选择LASSO来进行筛选，这是双重检验，我们以上的结果是否准确。Lasso回归复杂度调整的程度由参数lambda来控制，lambda越大模型复杂度的惩罚力度越大，从而获得一个较少变量的模型。**

install.packages("glmnet")

library(glmnet)

**## lasso model ，同样从一次项到多次项依次建模**

f1 <- as.formula(Y~ .)

x1 <- model.matrix(f1, dat)

cvfit1 <- cv.glmnet(x1, dat$Y)

c1 <- coef(cvfit1, s = "lambda.1se")

c1

#E1, E6, G9

f2 <- as.formula(Y~ .^2)

x2 <- model.matrix(f2, dat)

cvfit2 <- cv.glmnet(x2, dat$Y)

c2 <- coef(cvfit2, s = "lambda.1se") #G9, E1:E6, E1:G9, E6:G9

f3 <- as.formula(Y~ .^3)

x3 <- model.matrix(f3, dat)

cvfit3 <- cv.glmnet(x3, dat$Y)

c3 <- coef(cvfit3, s = "lambda.1se") #G9, E1:E6, E1:G9, E1:E6:G9

sink('c3.txt')

c3

sink(NULL)

f4 <- as.formula(Y~ .^4)

x4 <- model.matrix(f4, dat)

cvfit4 <- cv.glmnet(x4, dat$Y)

c4 <- coef(cvfit4, s = "lambda.1se") #G9, E1:E6, E1:G9, E1:E6:G9

sink('c4.txt')

c4

sink(NULL)

**## 综合以上所有lasso显著的结果，我们建立新的线性回归模型，进行再筛选**

fit7 <- lm(Y ~ E1 + E6 + G9 + E1:E6 + E1:G9 + E6:G9 + E1:E6:G9,data=dat1)

summary(fit7)

# G9, E1:E6 signiificant 这里就是我们lasso筛选的最终结果，我们发现两种方法结果一致

# Adjusted R-squared = 0.4507

BIC(fit7) # 73062.45

##########################################################################

**### check assumptions ###**

**#得到最终的模型后，我们还需要对模型进行检验，线性回归最重要的是残差要有等方差，符合正态分布等特点。**

**#画出我们最终选择的模型结果**

par(mfrow=c(2,2))

plot(fit4)

**#检验模型残差**

install.packages("car")

library(car)

**#检验残差是否彼此独立**，**p-value 大于0.1，符合彼此独立的条件，符合线性回归**

durbinWatsonTest(fit4) #p-value = 0.266

**#检验残差是否符合正态分布，p-value 大于0.1，是正态分布符合线性回归**

shapiro.test(fit4$residuals) #p-value = 0.2232

**#检验残差是否符合齐方差性，p-value 大于0.1，是齐差，符合线性回归**

ncvTest(fit4) #p-value = 0.4089509

**#如果以上检测不通过，不符合线性回归，那我们就需要做BOX-COX**

##########################################################################

# boxcox

##########################################################################

**#引入MASS包**

library(MASS)

**#引入MASS包**

**Fit1 <- lm(Y~.,data=dat) 画出fit1，我们可以看残差是否成喇叭型状，如果是，需要BOX-COX**

**转换**

plot(fitted(fit1), resid(fit1),

cex=1.2, pch=21, col="red", bg="orange",

xlab="Fitted Value", ylab="Residuals")

**#找出最适合的lamba值，才能作出最好的BOX-COX转换**

b=boxcox(fit1)

I=which(b$y==max(b$y))

b$x[I] #lambda=0.83

**#找到最好的lamba,此时我们要通过lamba改变Y使其变成Ylam,这样我们就进行了BOC-COX转换**

lambda<- 0.83; Ylam<- (Y^lambda-1)/lambda

lm.lam<-lm(Ylam~.); summary(lm.lam)

**#最后我们就重复以上的步骤，依次从一次项到多次项，进行多次建模，得到最终的结果。**