一、变量的基本定义和基础操作

1. 数值型变量的赋值

a = 5

2. 向量赋值

x = c(1:6) , c()为生成向量对应的函数

3. 向量中元素的访问

x = c(1:6)

x[3] , 中括号中的数字代表所访问的数值在向量x中的位置。

x[-3], 负数的标度表示取补集，即返回向量x中除第3位以外的其他元素。

4. 矩阵的定义

B =matrix(c(1:10),nrow=2,ncol=5,byrow=TRUE), matrix()未定义矩阵的函数，括号中第一个位置为写入矩阵中的元素，nrow参数位行数，ncol参数位列数，byrow=TRUE，表示数据按行的顺序书写。

5. 矩阵元素的访问

B[1,] 访问矩阵中的第一行

B[,2] 访问矩阵中的第二列

B[2,1] 访问矩阵第二行第一列的元素

B[,2:5] 访问矩阵2到5列的元素

B[,-4] 访问矩阵中除第4列的元素

6. 常用统计函数

sum() 求括号中对象的各个元素和

mean() 求括号中对象元素的均值

max() 求括号中对象元素中的最大值

min() 求括号中对象元素中的最小值

7. 其他矩阵信息的提取

dim(B) 返回矩阵的维度，第一个值为行数，第二个值为列数

dim(B)[1] 可访问矩阵的行数

length(B) 返回对象的长度，（请自行测试返回值是行还是列）

二、数据分析案例相关

1. 读取数据源

d = read.csv(“PRSA\_data\_2010.1.1-2014.12.31.csv”，header=TRUE) read.csv为读取csv文件的函数，使用该函数前，请将工作路径设置在数据源所在文件夹下。函数中第一个位置是文件名，需要写明扩展名，并且用双引号。header为控制是否读取表头的参数，如果原文件中有表头用TRUE，无表头用FALSE.

2. 去掉反应变量中空值

d1 = d[!is.na(d$pm2.5),]， “$”用来访问某对象对应的子属性，“d$pm2.5”表示读取数据源d中名称为pm2.5的列。Is.na() 函数将返回一个逻辑半段的向量，括号中对象的元素为空的返回TRUE，不为空的返回FALSE。“!”表示取反。“!is.na(d$pm2.5)”出现在数据源第一个位置，称为筛选记录的条件判别。

3. 数据源表头名称的访问

names(d1) 将返回d1中的各个列的表头，结果为一个向量。

names(d1)[3] 访问表头中的第3个名称。

4. 数据源的基本信息提取。

summary(d1) , summary()函数可以返回括号中对象的总结，返回内容会根据对象的不同进行调整。summary(d1)将返回所有变量的信息，其中连续变量返回最大值，最小值，均值等统计结果。分类变量将返回各类的频数。

5. 线性回归的基本形式

lr = glm(pm2.5~., data=d1)， glm()函数是一般线性模型求解释使用的函数，其中的参数“family“” 会对回归类型进行限制，默认状态下为线性回归。glm()函数第一个位置为表达式，其中“~”左侧是反应变量，即y，右侧为自变量，即x. “.”表示出了pm2.5以外的变量皆为自变量，“data=d1”指明数据源。

6. 线性回归中自变量的形式变化

lr2 =glm(pm2.5~No,data=d1) 表示仅以“No”作为自变量

lr2 =glm(pm2.5~No+TEMP,data=d1) 表示以“No”和“TEMP”为自变量

lr2 =glm(pm2.5~No+I(No^3) ,data=d1) 表示以“No”和“No”的3次方为自变量

lr2 =glm(pm2.5~poly(No,5) ,data=d1) 表示以”No”的1到5次方为变量。

7. 线性回归的结果

summary(lr) 将反回线性回归结果“lr”中包含的信息，如参数的估计值，标准误差，t检验结果，p-value等。

lr$coefficients 将返回对应的各个参数的估计值。”$”符号支持返回”lr”对应的多种返回值，除了” coefficients”外，其他返回值可以通过?glm查询。

8. 训练集与测试集的划分

为了对比不同方法的性能，通常将所获得的数据分为训练集和测试集两个部分，用训练集建模和参数优化，用测试集来进行比较。训练集和测试集的划分，可以通过矩阵的条件筛选完成

test = d1[d1$year==2014,] 表示取出d1中year为2014的数据

train = d1[d1$year<=2013,]，表示取出d1中year小于等于2013的数据

9. 使用线性回归模型进行预测

lr3 =glm(pm2.5~.,data=train)

pre = predict(lr3,newdata=test)， predict()函数将对已经生成的模型对象进行预测，其中第一个位置是已经得到的模型，”newdata=”是指明预测所使用的的新数据集。请注意一定保证“train”和“test”的结构完全一致。

10. 训练集数据的交叉验证

交叉验证需要将训练集中的数据分成不交叉的若干份，以最常见的10份为例。计算中，将以此使用这10份数据中的一份作为验证，其余9份建立模型，共计10次线性回归。所以首先要确定随机的10份数据如何分出。我们采用随机数生成，来为train中的数据添加一个标识其所属数据份数的量“label”.具体如下

label= sample(c(1:10),dim(train)[1],replace=TRUE)

d3 = cbind(train,label)

sample()函数将抽取随机数，第一个位置是抽取范围，第二个位置是抽取次数，replace=TRUE表示有放回。cbind()函数负责数据按列进行拼接。

11. 计算以第一组为验证，其余为建模数据时的误差

lr5 =glm(pm2.5~.,data=d3[d3$label!=1,]) 使用d3中label不为1的数据组为数据源进行回归。

pre2 =predict(lr5,newdata= d3[d3$label==1,]) 使用d3中label为1的数据组来产生预测值

error=pre2-d3[d3$label==1,]$pm2.5 计算预测值和真实值的差异

mse =sum(error^2)/length(pre2) 计算误差mse

12. 计算交叉验证误差的均值（以10组中的一组为验证，其余建模，循环10次）

mse1 = c()

for(i in 1:10)

{

lr6 = glm(pm2.5~No,data=d3[d3$label!=i,]) (假设模型中只包含变量“No”)

pre3 = predict(lr6,newdata= d3[d3$label==i,])

error=pre3-d3[d3$label==i,]$pm2.5

mse1[i]= sum(error^2)/length(pre3)

}

mean(mse1)

13. 考量模型中变量”No”的指数从1到15变化的15个模型，计算这15个模型对应的交叉验证误差的均值

mse2 = matrix(rep(0,150),nrow=10)

for (j in 1:15)

{

for(i in 1:10)

{

lr7 = glm(pm2.5~poly(No,j),data=d3[d3$label!=i,]) (j为模型中变量“No”的指数)

pre4 = predict(lr7,newdata= d3[d3$label==i,])

error=pre4-d3[d3$label==i,]$pm2.5

mse2[i,j]= sum(error^2)/length(pre4)

}

}

mmse1 = apply(mse2, 2, mean)

plot(mmse1)

lines(mmse1)

14. 考量模型中变量”No”的指数从1到15“TEMP”的指数从1到10变化的150个模型，计算这150个模型对应的交叉验证误差的均值

mse3 = matrix(rep(0,1500),nrow=10)

for (k in 1:10)

{

for (j in 1:15)

{

for(i in 1:10)

{

lr8 = glm(pm2.5~poly(No,j)+ploy(TEMP,k),data=d3[d3$label!=i,]) (j为模型中变量“No”的指数,k为模型中变量“TEMP”的指数)

pre5 = predict(lr8,newdata= d3[d3$label==i,])

error=pre5-d3[d3$label==i,]$pm2.5

mse3[i,15\*(k-1)+j]= sum(error^2)/length(pre5) (列标15\*(k-1)+j对应存储的是模型变量指数为k和j时的结果。由于j的变化是从1到15，所以(k-1)前的系数为15)

}

}

}

mmse2 = apply(mse3, 2, mean)

15. 根据价差验证误差选取最终模型，并计算测试集中的预测误差（以13中的例子为参考）

power = whichi.min(mmse1)

lrf = glm(pm2.5~poly(No,power),data=train)

pref = predict(lrf, newdata=test)

errorf = pref-test$pm2.5

msef = sum(errorf^2)/length(pref)

二、其他应用

1. 在微信红包中，计算抽取红包位置的收益平均值

number = 10000 （红包的发放次数）

bonus = 20 (每次发红包的金额)

pack = 40 (每次发红包的个数)

x = matrix(rep(0,pack\*number),nrow=pack) （储存每次红包在每个位置的金额）

for (j in 1:number)

{

x[1,j]=max(0.01,round(runif(1,min=0.01,max=bonus/pack\*2),2)-0.01) （第一个抽取红包的金额）

for (i in 1:38)

{

ulimit= (bonus-sum(x[,j]))/(pack-i)\*2 （计算第j次红包，第i个位置的金额上限）

x[i+1,j]=max(0.01,round(runif(1,min=0.01,max=ulimit),2)-0.01) （第i+1个位置，在第j次红包中抽取的金额）

}

x[pack,j]=bonus-sum(x[,j]) （最后一个位置抽取的红包）

}

final = apply(x,1,mean)

2. 在微信红包中，计算抽取红包的每个人的收益平均值。已知抽取红包的顺序是随机的，试计算40个人，抢40个总价值为20元红包，平均收益是多少。（如果有10000次红包可以抢）To be continued

number=10000

bonus=20

pack=40

x=matrix(rep(0,pack\*number),nrow = pack)

label=sample(c(1:40),40,replace = FALSE)

for (j in 1:number)

{

x[label=1,j]=max(0.01,round(runif(1,min = 0.01,max = bonus/pack\*2),2)-0.01)

for (i in 1:38)

{

ulimit=(bonus-sum(x[,j]))/(pack-i)\*2

x[i+1,j]=max(0.01,round(runif(1,min=0.01,max=ulimit),2)-0.01)

}

x[pack,j]=bonus-sum(x[,j])

}

number=10000

bonus=20

pack=40

x=matrix(rep(0,pack\*number),nrow = pack)

for (j in 1:number)

{

x[1,j]=max(0.01,round(runif(1,min = 0.01,max = bonus/pack\*2),2)-0.01)

for (i in 1:38)

{

ulimit=(bonus-sum(x[,j]))/(pack-i)\*2

x[i+1,j]=max(0.01,round(runif(1,min=0.01,max=ulimit),2)-0.01)

}

x[pack,j]=bonus-sum(x[,j])

x[,j]=sample(x[,j],40,replace=FALSE)

}

number=100

bonus=100

pack=40

x=matrix(rep(0,pack\*number),nrow = pack)

for (j in 1:number)

{

x[1,j]=max(0.01,round(runif(1,min = 0.01,max = bonus/pack\*2),2)-0.01)

for (i in 1:38)

{

ulimit=(bonus-sum(x[,j]))/(pack-i)\*2

x[i+1,j]=max(0.01,round(runif(1,min=0.01,max=ulimit),2)-0.01)

}

x[pack,j]=bonus-sum(x[,j])

}