回归模型

简单线性回归

```
data(anscomble)  # 加载数据集anscomble
attach(anscomble) # 将anscomble的列名加载到R环境中
cor(x1,y1)  # 协方差，如果有多列，则是返回协方差矩阵

yield.fit = lm(yield~content,data = anscomble) # 线性回归模型
abline(yield.fit,col = "red", lwd = 3)  #刻画回归线
summary(yield.fit)

yield.fit$residuals #残差
```

一些基本概念

total sum of squares : $SS_{tot} = \sum(y_i - \bar{y})^2$ 。 也就是样本的总方差

regression sum of squares: $SS_{reg} = \sum(\hat{y}_i - \bar{y})^2$. 也就是可解释方差

residual sum of squares: $SS_{res} = \sum(y_i - \hat{y}_i)^2$ 也就是残差

$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

`rsqrd <- 1- deviance(yield.fit) / sum((yield-mean(yield))^2)`

$adjusted R^2 = 1 - \frac{(1-R^2)(N-1)}{(N-p-1)}$

N是样本数量，p是变量数

线性回归的假设

1. x,y是线性关系
2. 误差没有自相关
3. 误差的方差相等
4. 变量之间无共线性
5. 误差服从正态分布。（没有离群值）

求标准化残差

在一元线性回归中，令矩阵A为

$$\begin{bmatrix} 1 & x_1 \\ \cdots & \cdots \\ 1 & x_n \end{bmatrix} \tag{1}$$

它有一个对应的hat matrix $H = X(X^TX)^{-1}X^T$

leverage：h_{ii}就是第H矩阵上对角线上的第i行的数据。

那么它对应的标准化残差是 $t_i = \frac{\hat{\epsilon}_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$

$\hat{\sigma}^2 = \frac{1}{n-m}\Sigma\hat{\epsilon}_j^2$ 就是方差的近似估计。m就是变量数目。

```
# 算法实现
X <-cbind(rep(1,17),snake$content)  # 共有17个样本
H <- X %*% solve(t(X)%*%X) %*%t(X)
h <-diag(H)
s <-sqrt(sum((residuals(yield.fit))^2)/df.residual(yield.fit))#计算方差
t <- residuals(yield.fit) / (s*sqrt(1-h))


# 直接获得
rstandard(yield.fit)
```

库克距离

用以评价删除一个数值点对回归的影响。一般观察cook值大于1 的数据点

计算 `cooks.distance(yield.fit)` #yield.fit就是线性模型

多元回归

基本操作

```
water = water[,-n] # 删除第n列
library(corplot)
cor(water) #协方差矩阵
```

特征选择

```
fit = lm(BSAAM~.,data  = water)  # ~. 意思是将数据集中其他所有列作为解释变量。

# 逐步递减删除变量
library(leaps)
sub.fit = regsubsets(BSAAM~.,data  = water)
summary = summary(sub.fit)
# 返回有1: n 个解释变量时的 R方, AIC,CP,BIC值。方便做比较
# 其中 R方越大越好, AIC,CP,BIC越小越好
plot(summary$cp,xlab = "numbers of features", ylab = "cp")
#通过上图选取特征。

best.fit =lm(BSAAM~APSLAKE+OPRC+OPSLAKE,data = water)
##检验共线性与异方差
#`vif`检验共线性, 有共线性的需要删除, 值大于5的就需要删除
#`bptest`检验异方差, 需要导入`lmtest`包
vif(best.fit)
library(lmtest)
bptest(fit.2)


#检验拟合效果
socal["actual"] = water $BSAAM
socal["forcast"] = NA #用na创建空列
socal["forcast"] = predict(fit.2)
library(ggplot2)
ggplot(socal,aes(x = forcast, y =actual))+geom_point()+geom_smooth(method= lm )
```

逻辑回归

实质上和普通回归没区别，只是将y的值从负无穷到正无穷，变换到了0到1

以下是逻辑回归的一个例子

数据集描述

- ID: This is the sample code number
- V1: This is the thickness
- V2: This is the uniformity of the cell size
- V3: This is the uniformity of the cell shape
- V4: This is the marginal adhesion
- V5: This is the single epithelial cell size
- V6: This is the bare nucleus (16 observations are missing)
- V7: This is the bland chromatin
- V8: This is the normal nucleolus
- V9: This is the mitosis
- class: This is the tumor diagnosis benign or malignant

```
# 加载、修饰数据集
library(MASS)
data(biopsy)
str(biopsy)
dataset <-biopsy
dataset$ID = NULL
names(dataset) =
c("thick","u.size","u.shape","adhsn","s.size","nucl","chrom","n.nuc","mit","class")
# 重命名每列
dataset <- na.omit(dataset) # 删除具有na值的列

bc = cor(dataset[,1:9])
corrplot.mixed(bc)
#观察协方差

##########划分数据集
set.seed(123) # 设置随机种子，确保每次随机结果是一样的
ind = sample(2,nrow(dataset) ,replace = TRUE , prob=c(0.7,0.3))#产生随机数，用以索引。将数据集中70%的数据作为训练集。
train = dataset[ind==1,]
test = dataset[ind == 2,]

########################回归
```

```
fullfit = glm(class~.,family = "binomial",data = train)
print('confint_____')
print(confint(fullfit)) #置信区间
print("end_____")
print(vif(fullfit))
#######################

##################训练集正确率
train$prob = predict(fullfit,type = "response")
contrasts(train$class)
train$predict  = rep("benign",dim(train)[1])
train$predict[train$prob>0.5] = "malignant"
mean(train$predict == train$class)
##############

#测试集正确率
test$prob = predict(full.fit,newdata = test , type="response")
#将测试集数据喂入模型中
test$predict = rep(”benign”, 209)
test$predict[test$ prob > 0.5]=”malignant”
table(test$predict, test$class)
mean(test$predict == test$class)
```

金融模型

收益率计算

毛收益率（gross return)

$r_g = \frac{s_t}{s_{t-1}}$

简单收益率

$r_t = \frac{s_t - s_{t-1}}{s_{t-1}}$

对数收益率

$r_{log} = \log(1 + r_t)$

```
library(quantmod)
AA <-getSymbols("AAPL",source = "YAHOO",from = "2015-01-01",to = "2018-01-01")
library(tseries)
res <- get.hist.quote(instrument= "AAPL",start="2015-01-01",end="2018-01-01")
# 获取数据的方法

#将dataframe写入文件
data1 <- as.data.frame(AAPL)
write.csv(data1,file= "AAPL.csv")

# 计算收益率
AAPL.close <-as.numeric(res$Close)
gross.return <-AAPL.close[2:length(AAPL.close)] / AAPL.close[1:(length(AAPL.close)-1)]
```

```
simple.return <-diff(AAPL.close) / AAPL.close[1:(length(AAPL.close)-1)]
log.return <- diff(log(AAPL.close))
```

债券价值

```
# P是票面，C是每期利息，r是年利率，t是已付息年限，T是总年限,k是每年付息次数

# 不包含已支付利息
BV_prime<-function(P,C,r,t,T,k)
    {
    tmat <- T-t
    if(tmat!=0)
        {
        i <-seq(1,tmat*k)
        sum(C/(1+r/k)^i)+P/(1+r/k)^(tmat*k)
    }else
        P/(1+r/k)^(tmat*k)
}

# 包含已支付利息
BV <- function(P,C,r,t,T,k)
    {
    tmat <- T-t
    arcued <- C*k*t # 已付利息，在该算法下，已付利息不复利
    if(tmat!=0){
        i<-seq(1,k*tmat)
        arcued+sum(C/(1+r/k)^i)+P/(1+r/k)^(k*tmat)
    }else
        arcued+P/(1+r/k)^(k*tmat)
}


#计算多期债券价值，且利率浮动
# 利率浮动时

#生成利率
rvec <-round(c(r,r+rnorm(T)*0.005),4)  # round表示四舍五入4位

simbv <-function(P,C,rvec,T,k)
    {
    BVvec = rep(0,T+1)
    for(t in 0:T){
        i<-t+1
        BVvec[i] <-BV(P,C,rvec[i],t,T)
    }
    BVvec
}
```

模拟股票价值

使用高斯分布

```
T <-45 # days
Svec <-round(c(1,1+1.1*rnorm(T)*0.0025),4)
plot(Svec,type="l")
```

使用布朗运动

dt = $t/steps$

nuT = $(\mu - \sigma^2/2) * dt$

sigmaT = $\sqrt{dt} * \sigma$

$s_1 = s_0 * \exp rnorm(1, nuT, sigmaT)$

PPT上代码和文字不一样，以代码为准

```
# s0 是初始值，mu是均值，sigma是方差，T是总时间， repl是路径数目
simGBM <-function(S0,mu,sigma,T,numsteps,numrepl)
    {
    dt <-T/numsteps
    nuT <- (mu-sigma^2/2)*dt
    sigmaT <- sqrt(dt)*sigma
    pathMatrix = matrix(nrow = numrepl,ncol = numsteps+1)
    pathMatrix[,1] <-S0
    for(i in 1:numrepl)
        {
        for(j in 2:(numsteps+1))
            {
            pathMatrix[i,j] <-pathMatrix[i,j-1] *exp(rnorm(1,nuT,sigmaT))
        }
    }
    return(pathMatrix)
}
```

AR模型

$X_t = a_0 + a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + \epsilon_t$

其中ϵ_t服从正态分布

```
# AR 模型  模型为 xt = 6+0.5*xt-1+ epsilon
AR <- function(x0,a0,a,sigma,T)
    {
    X <-rep(0,T+1)
    for(t in 1:T)
        {
        X[t+1] <-a0+a*X[t]+rnorm(1,0,sigma)
    }
    plot(0:T,X,pch=2)
    lines(0:T,X)
}
```

CAPM模型

```
# capm 模型拟合 其实就是将收益率-rf之后，用apple和spx做回归
spx.log.returns <-diff(log(spx.close))
aapl.log.returns <-diff(log(aapl.close))  # close 为收盘价


rf <- 0.05
rf.day <-rf/365
appl.e <-aapl.log.returns-rf.day
spx.e<-spx.log.returns-rf.day
capm1 <-lm(appl.e~spx.e)
plot(appl.e,spx.e)
abline(capm1)
```

最小方差投资组合

夏普比率：$\frac{R_p - R_f}{\sigma_p}$

假如有n个风险资产，各自的权重为W(nx1),收益率为R(nx1),协方差矩阵为V(nxn)。这个资产组合的收益为$W^T R$,风险为$W^T V W$。 也就是说，限制条件为$W^T R = R_p, W^T e = 1$,目标是$\min W^T V W$ 而我们可以用r中的 `solve.QP()` 来解决这个二次规划问题。

Solve.QP()求解二次规划问题(Quadratic Programming Problem)，此函数是实现了Goldfarb 与Idnani (1982, 1983)给出的对偶求解方法。

$$min(-d^T b + 1/2 b^T D b)$$

或者

$$min(1/2 b^T D b - d^T b)$$

限制条件$A^T b >= b_0$，注意：这里b_0是向量。后面那个$-d^T b$也许是没有的，取决于具体优化条件。具体到我们这里的均值方差投资组合优化问题，那么矩阵D可以认为是协方差矩阵，向量b是我们要求的资产权重向量。根据我们的假设后面那项$-d^T b$在这个问题中就是0。

也就是说该模型实际是就是将资产组合理论中的**W,V,R,E**做一些变换，用以**套用**在函数`solve.QP`上。

`solve.QP(Dmat,dvec,Amat,bvec,meq=0)`

其中，W作为输出，也就是对应向量 `b` 最优权重。

`solve.QP`函数参数中

V 与 `Dmat` 对应。R和E则是被整合到 `Amat` 中。 `bvec` 也要做对应修改。因为限制条件为$A^T * b >= bvec$

而 `solve.QP` 还有个参数 `dvec` 在这个模型中始终为0向量。

```
# 两个资产组合的情况，没太大意义
#就是已知两个资产的协方差、方差，带入求解。 不需要优化
muX <-0.06
muY <-0.15
rf <- 0.03
sigmaX <-0.4
sigmaY <-0.3
sigmaXY <-0.013
wX <-seq(0.1,0.1)
muP <- numeric(length(wX))
sigmaP <- numeric(length(wX))
sharp_ratio <-numeric(length(wX))
for(i in 1:length(wX)){
    muP[i] <- muX*wX[i] +muY*(1-wX[i])
    sigmaP[i] <- sqrt(wX[i]^2*sigmaX^2+(1-wX[i]^2)*sigmaY^2+2*wX[i]*(1-wX[i])*sigmaXY)
    sharp_ratio[i] <-(muP[i]-rf)/sigmaP[i]
}

# 多个资产组合 ， 看一下参数怎么设置的就ok
Dmat <- cov(portfolio_nodate)
Dmat <- Dmat * 252
Dim <- dim(Dmat)
D <-Dim[1]
dvec <- rep(0,Dim[1])
mu_vec <- sapply(portfolio_nodate, mean)
mu_vec <- mu vec * 252
Amat <- cbind(rep(1,D), diag(1, nrow = D))
bvec <- c(1, rep(0, D))
meq <- 1
library(quadprog)
QP <- solve.QP(Dmat = 2 *Dmat, dvec = dvec, Amat = Amat, bvec = bvec, meq=0)
QP$solution
QP
meq <- 2
aset <- seq(0.0001,0.2,0.001)
sharp_ratio <- numeric(length(aset))
sigma_p <- numeric(length(aset))
mu_p <- numeric(length(aset))

for (i in 1:length(aset))
{
  bvec <- c(1, aset[i], rep(0, D))
  Amat <- cbind(rep(1,D), mu_vec, diag(1, nrow = D))
  flag <- FALSE
  tryCatch(
    {QP <- solve.QP(Dmat = 2 * Dmat, dvec = dvec, Amat = Amat, bvec = bvec, meq= meq)},
    error = function(cond) {
      flag <- TRUE
    }
```

```
  )
  if (flag)
    break
  print(QP$solution)
  sigma <- sqrt(QP$value)
  mu <- crossprod(mu_vec, QP$solution)
  sharp_ratio[i] <- mu/ sigma
  sigma_p[i] <- sigma
  mu_p[i] <- mu
}

sharp_max_index <- sharp_ratio == max(sharp_ratio)
mu_index = mu_p != 0
sigma_p <- sigma_p[mu_index]
mu_p <- mu_p[mu_index]

plot(sigma_p, mu_p, type = 'l')
points(sigma_p[sharp_max_index], mu_p[sharp_max_index], col = 4, cex = 0.8)
text(sigma_p[sharp_max_index] + 0.009, mu_p[sharp_max_index],
```