**Week4**

set.seed(123)#随机种子生成器

data<-rnorm(1000,0,3)

hist(data)

head(data)

#该函数用于将一组数据分割成等宽的区间

equalwidth<-function(x,bin){

minx<-max(x)

maxx<-min(x)

width<-(maxx-minx)/bin

return(cut(x,breaks=seq(minx,maxx,width),include.lowest=TRUE))

}

##一个数据向量x和一个整数bin，表示你想要的区间数量。函数首先找到x中的最大值和最小值，然后计算每个区间的宽度。最后，

#它使用cut函数和seq函数来创建等宽的区间，并将x中的每个值分配到相应的区间。

transformdata<-equalwidth(data,4)

str(transformdata)

str(data)

table(transformdata)

barplot(table(transformdata))

head(transformdata)

discdata<-as.numeric(transformdata)#转化为数值型

#将数据分割成等频的区间

equalfreq<-function(x,bin){

n<-length(x)

freq<-n/bin

id<-c(1,seq(0,n-1,freq)[-1],n)

breaksid<-sort(x)[id]

return(cut(x,breaks=breaksid,include.lowest=TRUE))

}

transformdata<-equalfreq(data,4)

table(transformdata)

#k-means

#进行k-means 聚类分析

set.seed(123)

data<-rnorm(1000,0,3)

out<-kmeans(data,5)#k-means聚类分析，簇的数量为5

transformdata<-out$cluster#将聚类结果中的簇标签保存在变量中

#对簇标进行重新编码，以便簇的顺序和簇的中心顺序匹配

transformdatanew<-transformdata

transformdatanew[transformdata==3]<-1

transformdatanew[transformdata==1]<-2

transformdatanew[transformdata==4]<-3

transformdatanew[transformdata==5]<-4

transformdatanew[transformdata==2]<-5

#打印每个簇中的数据点数量

table(transformdata)

table(transformdatanew)

####supervised 卡方最小值离散化的两种方法

install.packages("discretization")

qchisq(p=0.1,1,lower.tail = FALSE)#查找卡方分布的分位数，上侧0.1分位数，自由度1

x<-c(1,3,7,8,9,11,23,37,39,45,46,59)#生成X向量

y<-c(1,2,1,1,1,2,2,1,2,1,1,1)#生成y向量

out<-discretization::chiM(cbind(x,as.factor(y)),alpha=0.1)#将x和y组合成一个矩阵，y转换为因子变量，矩阵进行卡方最小值离散化

transformdata<-out$Disc.data[,1]

x<-c(1,3,7,8,9,11,23,37,39,45,46,59)

y<-c(1,2,1,1,1,2,2,1,2,1,1,1)

out<-discretization::chi2(cbind(x,as.factor(y)),alp=0.5,del=0.001)#卡方最小值离散化

transformdata<-out$Disc.data[,1]

out

# 设置随机数生成的种子

set.seed(123)

# 生成一些随机数据

n <- 500

err <- rnorm(n, 2)

m <- function(x) { x \* sin(x) }

X <- rnorm(n, 3)

Y <- m(X) + err

# 绘制原始数据

plot(X, Y)

# 定义评估点

evalpoint <- seq(min(X), max(X), l = 100)

# 定义核函数（盒形核）

box <- function(evalpoint, x, h) {

sapply(x, function(x) { ifelse(((evalpoint - x) / h) >= -0.5 & ((evalpoint - x) / h) <= 0.5, 1, 0) })

}

# 进行核平滑估计

ksmoothbox <- function(x, X, Y, h) {

Kx <- box(x, X, h)

W <- Kx / rowSums(Kx)

return(W %\*% matrix(Y))

}

transformdata <- ksmoothbox(evalpoint, X, Y, 10)

# 绘制核平滑估计结果

plot(X, Y)

lines(evalpoint, transformdata, col = 2, lwd = 5)

# 查看核函数的维度

dim(box(evalpoint, X, 2))

# 检查条件

(evalpoint - x) / h >= -0.5

























