聚类算法---R的使用

1. 距离的计算和比较

In [ ]:

```
a<-matrix(runif(25),c(5,5))
a
```

In [ ]:

```
# 显示曼哈顿距离
dist(a,"manhattan")
```

In [ ]:

```
# 显示欧几里得距离
dist(a,p=2)
```

In [ ]:

```
a<-matrix(runif(15),c(3,5))
# 显示曼哈顿距离
dist(a,"manhattan")
```

In [ ]:

```
dist(a,p=2)
```

In [ ]:

```
a<-c(10,9,8)
b<-c(4,3,2)
c<-c(8,9,10)
```

a,b之间的余弦相似性

对于高维的文字向量，经常采用余弦相似性

In [ ]:

```
#a,b之间的余弦相似性
sum(a*b)/sqrt(sum(a^2)*sum(b^2))
```

In [ ]:

```
#a,c之间的余弦相似性
sum(a*c)/sqrt(sum(a^2)*sum(c^2))
```

In [ ]:

```
#b,c之间的余弦相似性
sum(b*c)/sqrt(sum(b^2)*sum(c^2))
```

In [ ]:

```
x=rbind(a,b,c)
x
```

In [ ]:

```
#a,b,c之间的欧几里得距离
dist(x,p=2)
dist(x)
```

2. 啤酒的系统聚类分析

2.1 读入数据

In [ ]:

```
####读入数据####
beer=read.table("C:/Users/bff/Desktop/me-ppt/beer_data.txt",header=T,sep=" ")
beer=as.matrix(beer)
beer
#注意beer为字符串格式
```

In [ ]:

```
####描述分析####
#x=beer[,-1]
x=apply(beer[,2:5],2,as.numeric)
rownames(x)=beer[,1]
x
summary(x)
```

In [ ]:

```
#因为不同变量取值差别较大，故将数据标准化
xstd=scale(x)
summary(xstd)
```

2.2 计算欧氏距离

In [ ]:

```
d=dist(xstd)
```

2.3 用不同联接方法系统聚类

In [ ]:

```
hc1=hclust(d,"single")  #最短距离法
hc2=hclust(d,"complete")#最长距离法，R中默认的联接方法
hc3=hclust(d,"ward.D")  #ward法 即离差平方和的方法，反映样本之间的差异程度
hc4=hclust(d,"centroid")#重心法
```

2.4 绘制谱系图

In [ ]:

```
####绘制谱四种联结方法得到的谱系图####
par(mfrow=c(1,1))#设置画布，如可以绘制2*2的组图。
#par(family='STKaiti')#设置字体
plot(hc1,hang=-1)#最短距离法的谱系图
```

In [ ]:

```
plot(hc2,hang=-1)#最长距离法的谱系图
```

In [ ]:

```
plot(hc3,hang=-1)#ward法的谱系图
```

In [ ]:

```
plot(hc4,hang=-1)#重心法的谱系图
```

解读和分析： 使用系统聚类方法时，点与点之间的距离用欧式距离，类与类之间的连接使用了四种不同的

距离定义，得到的结果也不同，接下来要结合具体的问题进行分析。

3. 啤酒的k-mean聚类分析

In [ ]:

```
cl=kmeans(xstd,3,20) # 聚为3类，最大迭代次数20
```

3.1 聚类结果展示

In [ ]:

```
cl # 展示K均值聚类的主要结果
```

In [ ]:

```
#### 也可以单独展示部分结果####
cl$cluster #展示每个样本观测属于哪一类
```

In [ ]:

```
#par(family='STKaiti')#设置字体
plot(x,col=cl$cluster,pch=2,lwd=1) ### 用前两个变量并标注三个类
```

In [ ]:

```
#### 也可以单独展示部分结果####
cl$centers #展示每一类的中心
```

4. 其他算法，以及聚类结果评价-侧影统计量

本例使用iris数据，前4个变量（前4列）依次是花萼长度、花萼宽度、花瓣长度、花瓣宽度四个特征，

第5个变量是类别变量，有3个类别，依次是:山鸢尾、变色鸢尾、维吉尼亚鸢尾。

前4个变量是定量变量，第5个变量是定性变量。下面的代码中提取前四列作为聚类的自变量。

In [ ]:

```
head(iris)  #展示iris数据的5个变量和前几行样本观测值
#pairs(iris)
plot(iris)
```

4.1 k-means对iris进行聚类分析

In [ ]:

```
iris2<-iris[,1:4]
iris.kmeans<-kmeans(iris2,3)
iris.kmeans
```

In [ ]:

```
#用table函数查看分类结果情况
table(iris$Species,iris.kmeans$cluster)
```

In [ ]:

```
plot(iris2$Sepal.Length,iris2$Sepal.Width,col=iris.kmeans$cluster,pch="*")
points(iris.kmeans$centers,pch="X",cex=1.5,col=4)
```

4.2 使用k-mediod方法聚类分析

K-Mediods函数跟Kmeans函数基本类似，不同的是，Kmeans是选择簇中心来表示聚类簇，而K-Mediods选择靠近簇中心的对象来表示聚类簇。在含有离群点的情况，下K-Mediods的鲁棒性（稳定性）要更好。

基于中心点的划分算法PAM是K-Mediods中的经典算法，但是PAM很难扩展到较大数据集上，而Clara算法是对PAM算法的改进，他是在较大数据集中分为几个小数据集，分别进行PAM算法，并返回最好的聚类。因此在处理较大数据集的情况下CLARA算法要优于PAM算法，

在R的cluster包中的PAM和CLARA函数分别实现了上述两个算法，但是这两个函数都需要用户指定k值，即中心点的个数。fpc包中的pamk()函数提供了更加强大的算法，该函数不要求用户输入k值，而是自动调用pam或者clara来根据最优平均阴影宽度来估计聚类簇个数来划分数据集。

In [ ]:

```
data=iris[,1:4]#取数据的前四列
#install.packages("cluster",repos="https://mirror.lzu.edu.cn/CRAN/")
library(cluster)# 加载包
pam3=pam(data,3)#k-mediod聚类方法，pam算法
length(which(pam3$clustering==1))#看一下每类中都有多少个样本
```

In [ ]:

```
table(iris$Species,pam3$clustering)
```

In [ ]:

```
pam3$silinfo$avg.width#查看silhouette width均值
```

In [ ]:

```
si3=silhouette(pam3)#查看每个silhouette统计量
plot(pam3)#Silhouette  Plot
```

In [ ]:

```
pam4=pam(data,4)
plot(pam4)
```

4.3 基于Clara算法的聚类

In [ ]:

```
iris2.clara<-clara(iris2,3)
table(iris$Species,iris2.clara$clustering)

#             1  2  3
# setosa     50  0  0
# versicolor  0 48  2
# virginica   0 13 37

layout(matrix(c(1,2),1,2)) #每页显示两个图
plot(iris2.clara)
layout(matrix(1))
```

4.4 基于Pamk算法的聚类

In [ ]:

```
#install.packages("fpc",repos="https://mirror.lzu.edu.cn/CRAN/")
library(fpc)
iris2.pamk<-pamk(iris2)
table(iris2.pamk$pamobject$clustering,iris$Species)


layout(matrix(c(1,2),1,2)) #每页显示两个图
plot(iris2.pamk$pamobject)
layout(matrix(1))

#     setosa versicolor virginica
# 1     50          1         0
# 2      0         49        50

#通过上述分类结果可以看到，pam和calra算法分类结果基本类似，但是pamk将三类分为了两类。。

```

In [ ]:

```
dim(iris)#返回行列数

idx<-sample(1:dim(iris)[1],40)
iris3<-iris[idx,-5]
iris3
hc<-hclust(dist(iris3),method = "ave")  #注意hcluster里边传入的是dist返回值对象

plot(hc,hang=-1,labels=iris$Species[idx])  #这里的hang=-1使得树的节点在下方对齐
#将树分为3块
rect.hclust(hc,k=3)
```

In [ ]:

```
groups<-cutree(hc,k=3)
groups

```

5. DBSCAN算法----基于密度的聚类

In [ ]:

```
#---基于密度的聚类分析，一种方法通过fpc包中的dbscan实现；另一种方法通过factoextra包实现
library(fpc)
iris2<-iris[-5]
ds<-dbscan(iris2,eps=0.42,MinPts = 5)
table(ds$cluster,iris$Species)

#打印出ds和iris2的聚类散点图
plot(ds,iris2)

```

In [ ]:

```
#打印出iris第一列和第四列为坐标轴的聚类结果
plot(ds,iris2[,c(1,4)])
```

In [ ]:

```
#另一个表示聚类结果的函数，plotcluster
plotcluster(iris2,ds$cluster)
```

我们使用factoextra包中的数据集multishapes进行演示dbscan

该包中的两个函数十分有用，一个用于确定最佳的簇数，一个用于可视化聚类的结果

In [ ]:

```
#install.packages("factoextra",repos="https://mirrors.tongji.edu.cn/CRAN/")
#remove.packages("factoextra")#卸载包

```

In [ ]:

```
#载入包
library(factoextra)
library(ggplot2)
data("multishapes")
df <- multishapes[, 1:2]
df0<-multishapes
df0$shape<-as.factor(df0$shape)
ggplot(df0,aes(x=x,y=y,colour=shape))+geom_point()

```

In [ ]:

```
#先用k-means算法得到初步聚类
set.seed(123)
km_result <- kmeans(df, 5, nstart = 25)
```

In [ ]:

```
fviz_cluster(km_result , df, geom = "point",
             ellipse= FALSE, show.clust.cent = FALSE,
             palette = "jco", ggtheme = theme_classic())
```

完全不是我们想要的结果。这样我们就回答了第一个问题，为什么需要基于密度的聚类算法，因为像这种奇奇怪怪形状的样本点，我们利用k-means算法根本无法将其进行聚类。那么基于密度的聚类算法为啥可以解决。大家看看这个图，每一类就是一些点密集地连接在一起。密集地也就是代表了密度的不同，所以这里大家就知道了，这个密度不是我们平时说的概率密度分布。实际上是一种样本密度的代表。

In [ ]:

```
fviz_cluster(km_result , df, geom = "point",
             ellipse= FALSE, show.clust.cent = FALSE,
             palette = "jco", ggtheme = theme_classic())
```