# 深入洞察你的客户-客户画像聚类分析

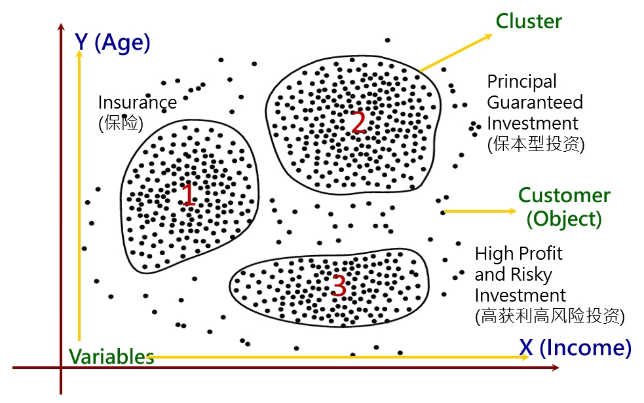
李御玺(leeys@mail.mcu.edu.tw)

## 聚类分析简介

* In cluster analysis, there is no pre-classified data. 在聚类分析中, 没有事先分类好的数据.
  1. 没有目标字段, 只有输入字段. 如想要从年龄和收入这两个方面把银行的客户进行聚类. 把年龄和收入相似的客户聚成一类, 年龄和收入都是输入字段, 并不需要给定一个目标字段. 只有在预测型的数据挖掘中才需要目标字段.
* Instead, cluster analysis is a process to group similar objects together. 聚类分析是把相似的对象聚合到一起的过程. 相似的对象, 即在输入属性上相似的对象.
  1. 聚类分析的目的: 根据数据的输入属性，将输入属性类似的数据聚在同一群
* All members in one cluster are similar to each other and different from the members of other clusters, according to some similarity metric
  1. 聚类的指导原则: 同一集群内成员的相似性要愈高愈好, 即同一集群内成员输入属性的相似性要愈高愈好，而不同集群间成员的相异性则要愈高愈好. 同一群的越像越好, 不同群之间的则差异性越大越好.

通过客户的输入属性把客户聚为一群一群的, 每一群客户都可以去描述它们的特征, 也就是针对每一群客户的客户画像. 有了不同群体客户的客户画像, 就能够有针对性的对这些有不同特征的客户采取不同的营销活动.

## 聚类分析的应用：银行客户聚类



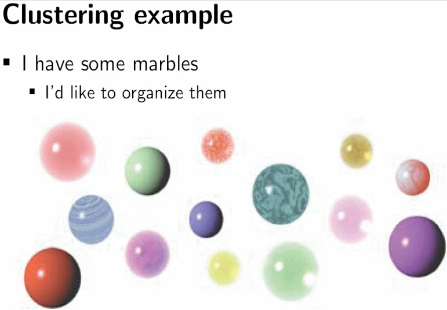
每一个点代表一个客户, 点聚集的比较密集的地方就是一个群体.

第1类客户推销保险, 身故或重残才能领到的保险.

根据一个群体的特征推销相应的金融产品. 这样营销成功的概率就会大在提高.

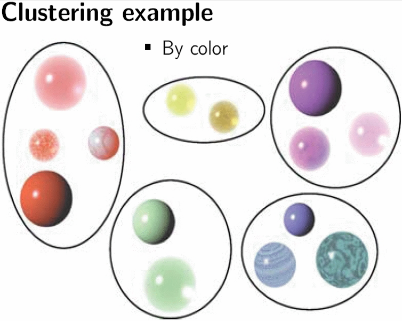
## Clustering Example

虽然聚类分析不需要目标字段, 但自己心中必须要有目标. 如对弹珠进行聚类分析时, 心中的目标可以是颜色, 大小, 透明度, 花纹样式等.

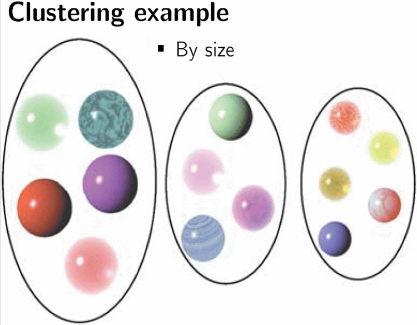


### By color

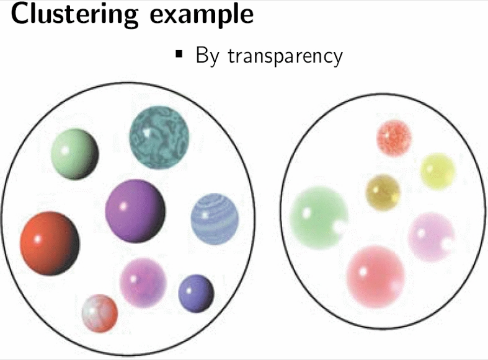
类似于收入字段



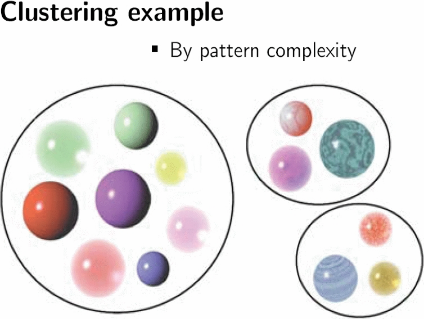
### By size



### By transparency



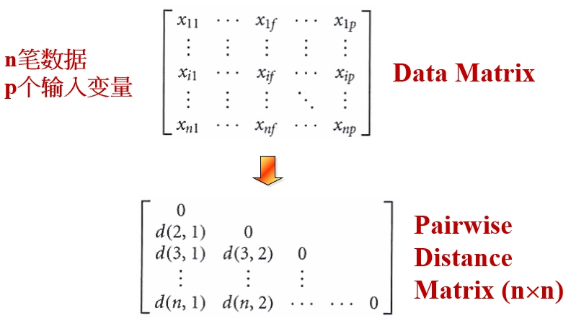
### By pattern complexity



## 聚类分析模型建置

* 聚类分析所关心的重点有三
  1. 如何以数字来表示成员间的相似性 (或相异性)？
  2. 如何根据这些相似性将类似的成员分在同一群？
  3. 所有成员分群完毕后，对每一群的特征应如何描述？

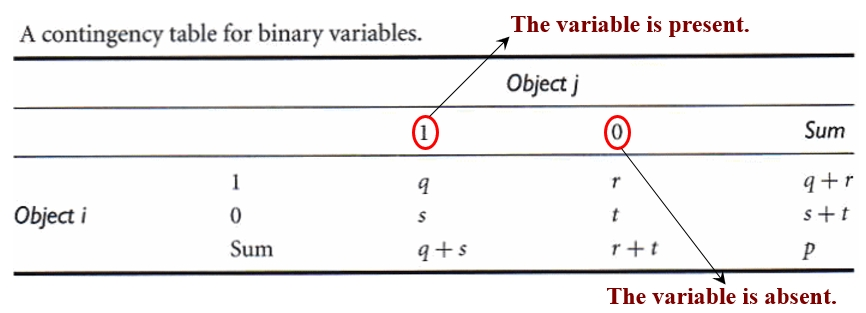
## 聚类分析：距离的计算



n个观测, p个变量, 相当于是n行p列的一个矩阵.

Pairwise Distence即计算两两之间的距离, 距离表示的就是相异性. 距离与相似性成反比, 距离越近, 相似性越高. d(2,1)为x21与x11之间的距离. d(n,1)为xn1与x11之间的距离, 以此类推. 只需要计算下半三角或上半三角的值即可. 是关于对称轴对称的.

### 输入属性都是二元变量的距离计算

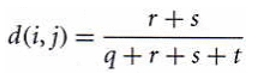


contingency table 为列联表, 对于二元的分类型变量, 计算得出列联表.

二元变量, 只能取两个值的变量, 如性别, 只能取"男", "女"两个值.

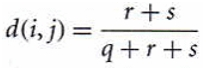
Object i, j 分别相当于第i, j行, 二元变量可以转换为因子, 它的值不是1就是0. q表示Object i = 1且Object j = 1的值共有q对. r表示Oi = 1, Oj = 0的值共有r对. 以此类推.

**简单匹配系数 Simple Matching Coefficient:**



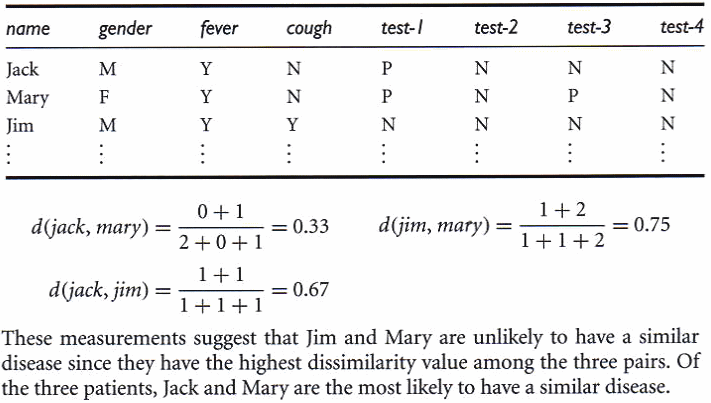
q+r+s+t为所有可能的组合数, r+s就是不一致的情况数, 不一致的情况数越少, 就意味着距离越小, 二者之间的相似性就越大.

**Jaccard Coefficient:**



对于JC来说, 去掉了分母中二者都为N的次数, 对于下面的例子来说, 二者都为N, 也可能说明是二者都没有去做对应的test, 所以这里使用JC系数. 即二者都为N, 并不一定说明二者相似.

#### 计算范例



对3个病人的病症和进行过的几个检查的结果进行分析, 计算病人之间的距离, 希望病症和检查结果相近的病人的距离较小.

主要是根据2个病症和4个test来进行计算.

列联表

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Mary | |
|  |  | Y | N |
| Jack | Y | 1+1=2 | 0 |
| N | 1 | 1+1+1=3 |

使用简单配对系数SMC时, d(jack, mary)=1/6

这里使用JC系数d(jack, mary) = 1/3

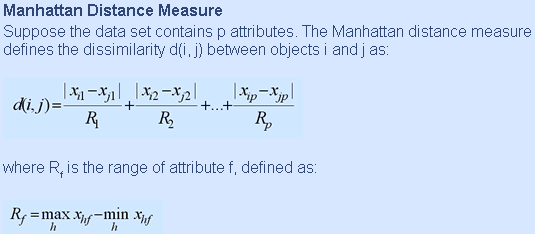
对于计算的结果, jack和mary的距离最短, 所以jack和mary最可能得了相似的疾病.

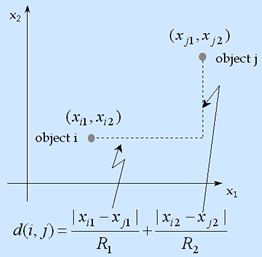
### 输入属性有不同类型变量的距离计算

* **Interval Variables 数值型的变量**
  1. An interval variable contains continuous measurements (e.g., height, weight, temperature, cost, etc.) which follow a linear scale. 数值型的变量包含着连续的观测值, 如高度, 重量, 温度, 成本等,
  2. It is essential that intervals keep the same importance throughout the scale. 在整个范围内, 都要保证是等间距的.
  3. **Nominal Variables 类别型变量**
  4. A nominal variable takes on more than two states. For example, the eye color of a person can be blue, brown, green or grey eyes. 类别型变量会取大于2的多个值. 如眼睛的颜色.
  5. These states may be coded as 1, 2, ..., M, however their order and the interval between any two states do not have any meaning. 类别型变量的状态可能会使用1, 2, …等数字来进行编码, 但是他们的顺序和两个状态之间的间隔并没有任何意义, 数字只是它的编码或表示方法.
  6. **Ordinal Variables 顺序型变量**
  7. An ordinal variable takes on more than two states. For example, you may ask someone to convey his/her appreciation of some paintings in terms of the following categories: 1=detest, 2=dislike, 3=indifferent, 4=like and 5=admire. 如对一幅画的喜好程度, 使用数字来表示不同的喜爱程度, 但这里的数字之间的距离, 从1到2, 以及从2到3, 等之间的距离不是固定的, 是因人而异的.
  8. In an ordinal variable, their states are ordered in a meaningful sequence. However, the interval between any two consecutive states are not equally distanced. 两个状态之间的距离并不是等距的.
  9. **Binary Variables 二元类别型变量**
  10. Binary variables have only two possible states. For example, the gender of a person is either female or male. 只能取两个值的类别型变量.

归纳起来, 可以分为2类, 数值型和类别型, 数值型变量和顺序型变量都可以归纳为数值型, 类别型和二元类别型则归为类别型. 在进行聚类分析时一般就使用2类的分类方法, 只分为数值型和类别型.

### 距离的计算-街区距离

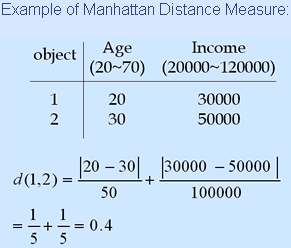




如果有两个变量, 第1个变量之间的距离加上第2个变量之间的距离, 就是两个变量之间的距离.



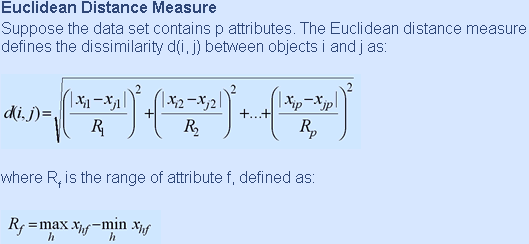
街区距离, 曼哈顿距离, 直角距离. 在一个个以方块分隔成的街区中, 想从一点到另一点, 所走的距离就是此直角距离.

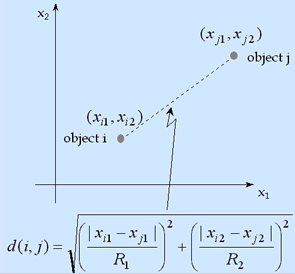


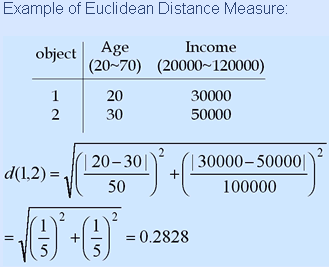
计算object1和object2之间的距离. 必须要先做标准化. 这里使用的是极差标准化. 两个对象之间某变量的差值除以这个变量取值的最大值与最小值之差.

### 距离的计算-欧几里得距离

即直线距离

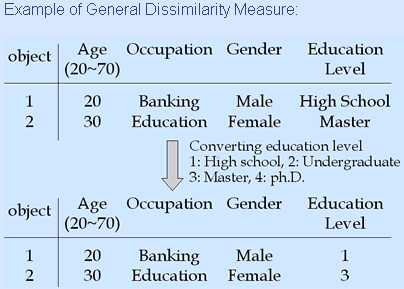






### 距离计算范例

不同类型数据的距离的计算

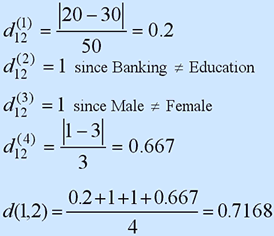


Age为数值型变量

Occupation 职业为类别型变量

Gender性别为二元类别型变量

Education Level 教育程度为顺序型变量. 高中为1, 本科为2, 研究生为3, 博士为4, 数字之间不等距. 把EL转换为数值型变量.



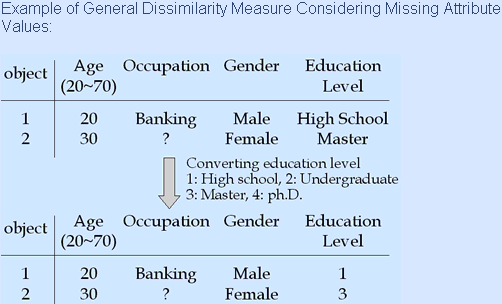
第二个维度职业之间的距离d(2), 职业为类别型变量, 值相同时为0, 值不同时为1.

第三个维度与第二个维度相似, 值为1.

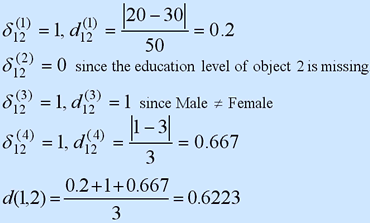
第四个维度按数值型变量进行处理.

总的距离为各个维度距离的均值, 这样得到的距离总是介于0和1之间. 取均值时就不会因为随着维度的增加或数据量的增加值无限增大的情况.

### 距离计算范例(空值处理)



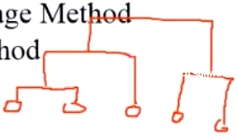
取各个维度的均值还可以在遇到空值时不进行填补. 因为职业中存在着空值, 就设置为0, 即不参与最终的运算.



## 聚类算法的分类

已经计算出来距离之后, 就要使用聚类算法进行分群

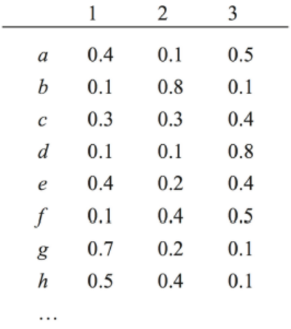
* Exclusive vs. Non-Exclusive (Overlapping). Exclusive即有排它性的聚类方法. 一个观测值只会属于某一个群体, 不会属于多个群. Non-Exclusive或Overlapping方法就是无排它性或可重叠的聚类方法. 一个资料点??可以属于多群, 只是概率不同而已. 一般来说, 非排它性的聚类方法是比较符合现实情况的, 但算法比较复杂, 运算速度比较慢, 并且在实际中操作起来比较困难. 如银行中把一个客户分到了多个群中, 如何去对这个客户进行针对性的推荐呢. 所以实际中用的几乎全都是排它性的方法.
* Hierarchical Methods vs. Partitioning Methods. 聚类方法的分类: 阶层型和分割型.
* Hierarchical Methods: SPSS, SAS. 阶层型分类方法大部分都是统计软件在提供.



实现过程, 有5笔数据, 先聚成5群, 从中计算距离最小的2群, 把这2群聚成1群, 再计算其余群中距离最小的, 聚群, 再从总体中取距离最小的, 聚群, 如此进行. 过程就是5,4,3,2,1, 阶层型一步步的进行聚群.

* Single Linkage Method & Complete Linkage Method. 单一链接法与完全链接法.
* Average Linkage Method & Centroid Method. 平均链接法, 中心法.
* Ward’s Method. 效果一般是最好的, 但聚类的速度较慢.
* Partitioning Methods. 分割式的聚类方法. 比阶层法的效果要好. 所以一般的统计软件都会提供分割式的聚类方法. 一般要提供把资料聚成多少群, 必须要事先决定群数. 一次性把要求的所有群聚合出来, 没有类似上面的5,4,3的过程.
* K-Means Methods: SAS EM为SAS数据挖掘工具, SPSS Modeler为SPSS数据挖掘工具. K-Means方法又快又好.
* K-Medoids Methods (PAM, …). 此算法的效果要比K-Means方法好, 但速度较慢.
* Kohonen Self-Organizing Maps (SOM): SPSS Modeler, SAS EM, 神经网络的聚类方法.
* TwoSteps: SPSS Modeler, SAS EM. 两步法, 是K-Means方法的变形. 第一步是决定群级数, 第二步是使用K-Means的方法进行聚类分析.
* …

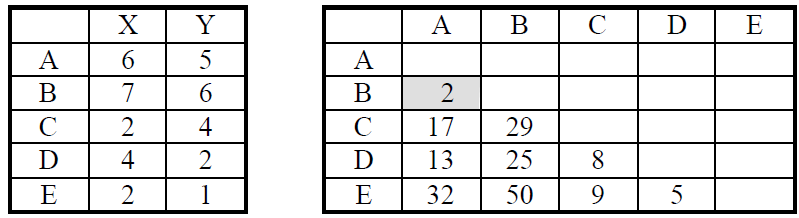
**Overlapping聚类方法**



把所有数据分成3群, a属于第1群的概率为0.4, 属于第2群的概率是0.1, 属于第3群的概率是0.5.

## 阶层式聚类法:

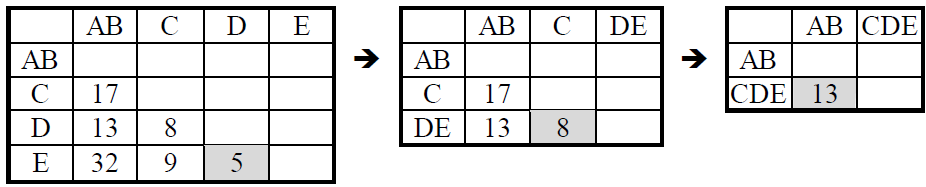
### 单一链结法(Single Linkage Method)



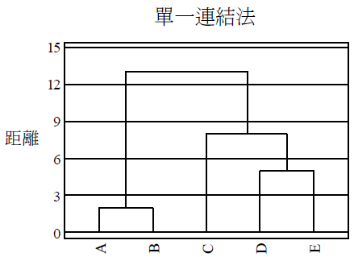
X, Y是两个输入变量, A-E代表5个数据点/5笔数据.

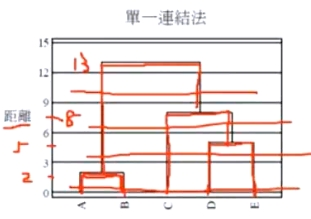
求出距离矩阵. 接近欧几里德的计算方式, 只是没有开根号. AB之间的距离为(6-7)^2+(5-6)^2=2. AC距离为(6-2)^2+(5-4)^2=17

单一连结法的操作步骤, 从距离矩阵下三角中找出距离的最小值. AB的距离最小, 就会首先把AB合并成一个群, 由5群变为4群, 再次计算距离矩阵, 如计算AB与C的距离时会把AB拆成A和B, 分别计算与C的距离, 然后再取距离中的最小值, 即把最近的距离作为AB与C这两个群之间的距离. 如此依次进行下去.



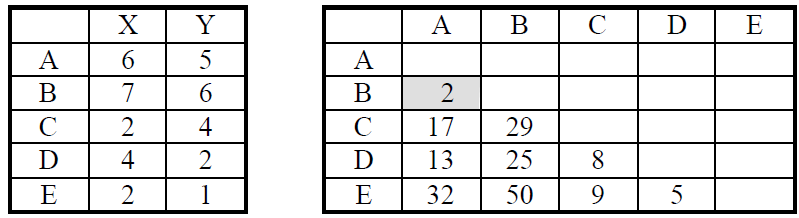
Dendrogram树形图, 描述阶层型聚类方法的聚类过程.



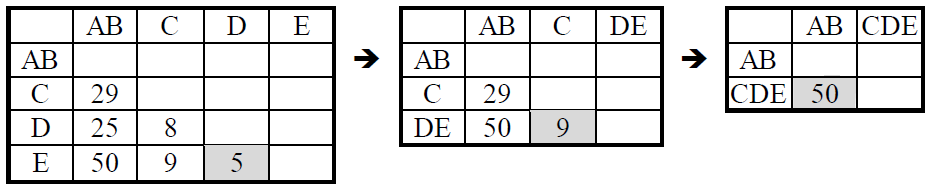


分别取1群, 2群, … 时的划分方法.

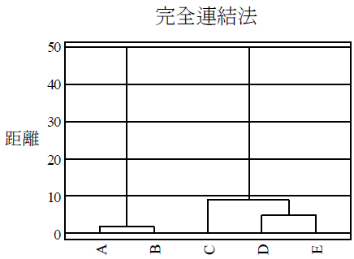
### 完全链结法(Complete Linkage Method)



先计算距离矩阵, 把距离最小的2类划分为一个群. 再次计算新的距离矩阵, 把AB与C之间的距离拆分为A与C和B与C之间的距离, 只是此时取二者中距离的最大值.

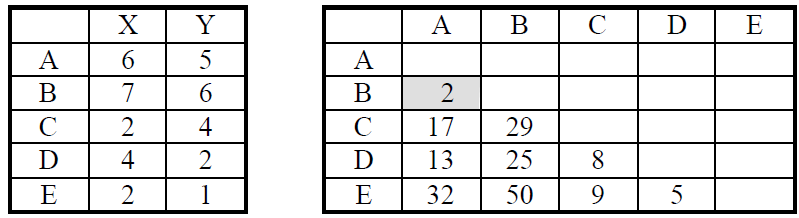


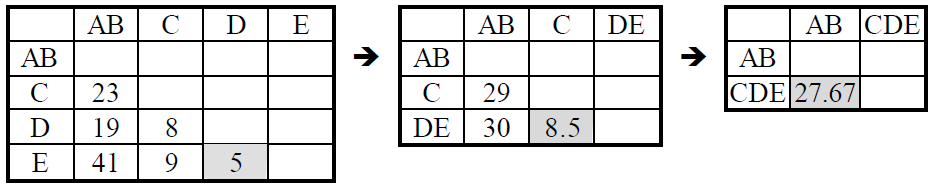
AB与DE之间的距离拆分为AB与D和AB与E之间的距离, 取最大值50. 然后把AB-C-DE在距离为50的地方聚合成一群.



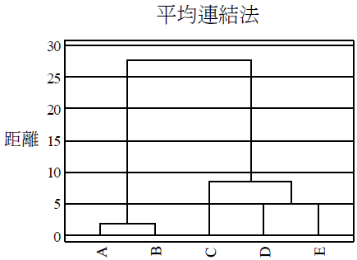
### 平均链结法(Average Linkage Method)

平均链结法取平均值.



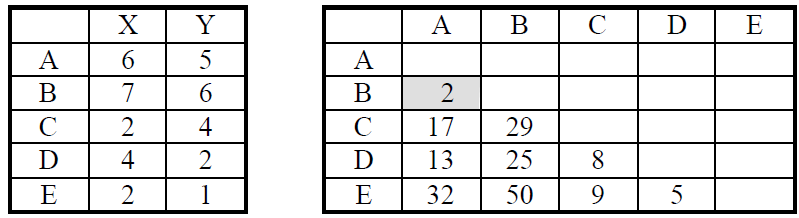


先计算AB之间的距离: (6-7)^2+(5-6)^2, 把AB聚成一类. 然后再次计算距离矩阵, AB与C的距离, 拆分为A与C和B与C的距离. 取AC, BC距离的平均值为AB与C的距离, 为(17+29)/2=23

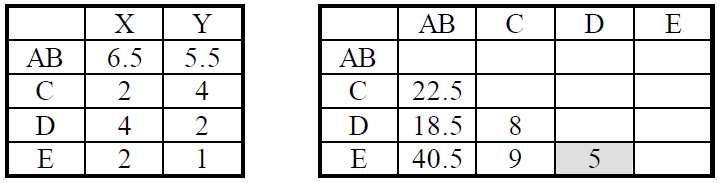


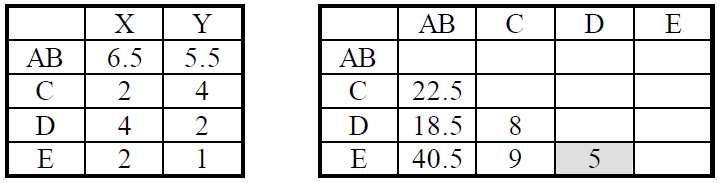
### 中心法(Centroid Method)

聚类之后会以群中心代表这个群. 再计算与各个群之间的距离.

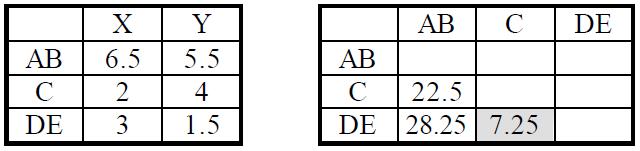


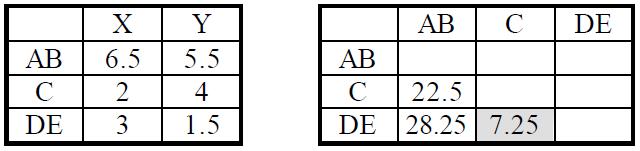
先求距离矩阵, 找到最小的距离并进行聚类, 如把AB聚为一类后, 是把AB当成一个新的变量, 再次计算距离矩阵时不再把它拆开. AB这个新变量的位置为原来AB位置的平均值, 即以AB群中心代表AB这个群. 再重新计算AB与C,D,E两两变量之间的距离.



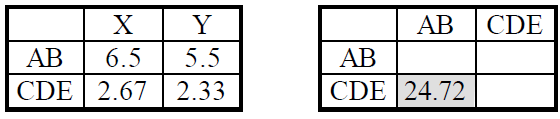


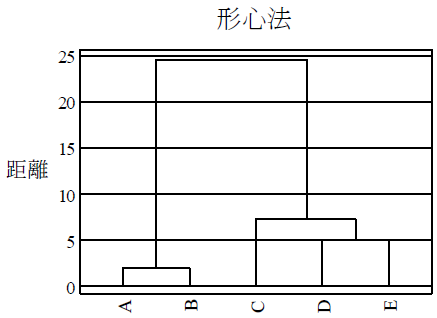
然后再把距离最小的两个群之间聚成一类. 如把DE聚成一类. 再重新计算距离矩阵, 如此进行.





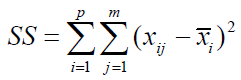
注意, 此时的CDE的位置不是使用C与DE计算得到的, 而是使用原始数据中的C, D, E来进行计算. 也就是求群中心时是所有的数据点都参与运算的, 此时不能把AB, DE看成一个整体. 即求位置时使用原始数据, 求距离矩阵时把聚的类看成一个整体. 即基本的思想是使用聚类后的群中心来计算与其它群之间的距离.



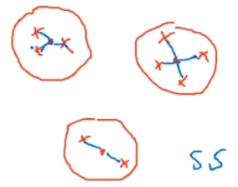


### 华德法(Ward’s Method)

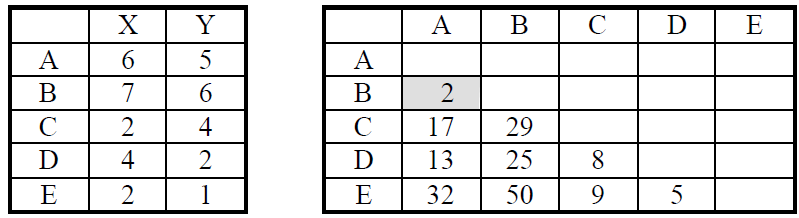
是阶层法中效果最好的, 也是运算速度最慢的方法.



通过计算SS值来判断一个聚类效果好不好. SS即每个资料点与群中心距离的平方和. SS值越小, 说明聚类的效果越好, 因为此时每个数据都更加接近于群中心. 华德法就是使用SS值来判断是将哪两个变量合在一起. 如果两个变量聚合到一起时得到的SS值是最小的, 就是最好的合并数据的方法. 有点类似作弊的方法, 就是使用评估聚类好不好的指标来当成是否把数据合并起来的指标, 所以得到的结果是最好的, 但是组合数是非常多的. 所以速度是最慢的.



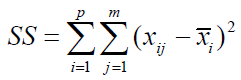
如把资料聚成3个群, 则SS等于3个群中每个点到各自的群中心的距离的平方和, 共9个平方和.



刚开始时因为没有聚类, 就没有群中心, 所以SS值为0. 假设先把AB合在一起, 计算AB的群中心, 为(6.5, 5.5). 求得SS=1. 而C,D,E自成一群, 群中心为它们自身, 计算得到的SS都为0, 所以把AB聚到一起时得到的总的SS为1. 再假定把CD合并在一起, 计算群中心和对应的SS值, 结果为SS=4, AB聚成一群时SS值最小, 所以要把AB聚到一起.

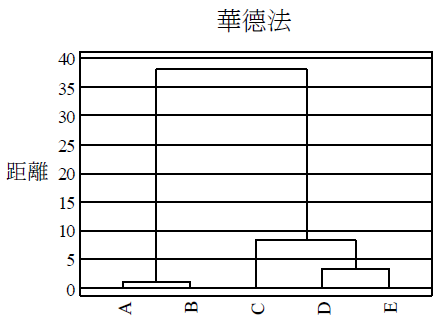
以前述的资料为例, 一开始如果将AB合并成一个集群, 则此时, 如果将CD合并成一个集群, 则 .

当合并成三个集群时, 如果是AB, CD, 及D的组合, 则联合组内SS就等于1+4+0=5.

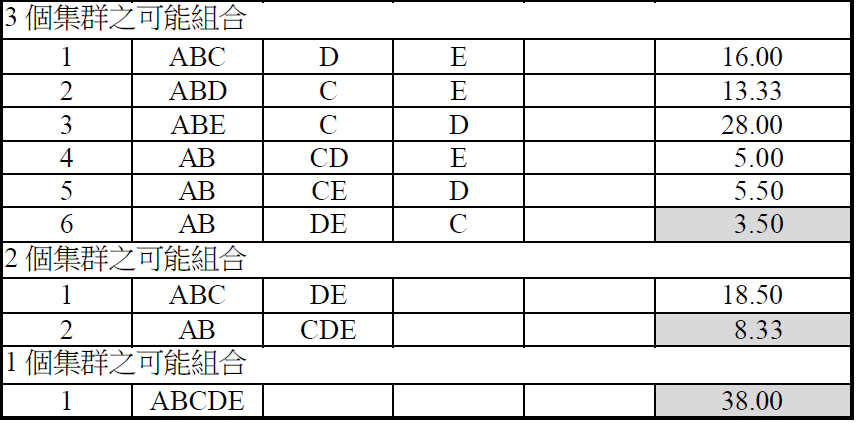




共有C(5,2)=10种组合. 由5群合并成4群时只有当AB合在一起时SS值才是最小的, 所以取AB的组合.



群数越少时, 更多的数据合并到一个群中, 群内差异就会变大, 群内元素距离群中心的距离会越来越大, SS值就是衡量群内差异的, 得到的SS值一定会越来越大.



### 总结

* 在大部分的研究中，以使用平均链接法及华德法较佳，而以单一连结法较差
* Partitioning Methods又优于Hierarchical Methods. 分隔式的聚类方法又优于阶层式的聚类方法.
* 阶层式的聚类方法不太适合数据量非常大的情况.
* 分隔式的聚类方法必须要事先确定聚类的群数. 所以就有了两阶段的聚类方法, 先确定群数, 再进行聚类.
* TwoSteps
* 决定群数
* 聚类

### 软件实现

#### 阶层法对iris数据进行聚类分析

# 需求分析: 150朵iris的数据, Species为3种. 使用Sepal.Length, Sepal.Width, Petal.Length, Petal.Width这4个变量对iris进行聚类. 看聚类的结果是否与Species中已经分好的类别是否相同或相近

iris2 <- iris[,-5]

#Sample 40 Records, 取出40个数据, 150个数据画的图太大了.

# 从150个数据中随机抽取40个数据进行聚类分析

index <- sample(1:nrow(iris2),40)

irissample <- iris2[index,]

# dist计算两两成对的距离矩阵, dist表示distence.

dist(irissample)

?hclust

#dist函數 distance matrix compute. 使用hclust函数来完成阶层式聚类的计算. method= "ward.D2"是使用修正后的方法, 这种方法的效果一般是最好的. method可以选用的方法有 华德法"ward.D", 改进的华德法"ward.D2", 单一链结法 "single", 完全链结法"complete", 平均链结法"average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or 中心法"centroid" (= UPGMC).

hclust.result <- hclust(dist(irissample), method= "ward.D2")

hclust.result

Call:

hclust(d = dist(irissample), method = "ward.D2")

Cluster method : ward.D2

Distance : euclidean

Number of objects: 40

# 画图, labels中使用原目标字段的值, 一共有40个值. 大部分同类的数据都会聚在一起, 只存在少量的重复. 当聚类的群小于等于3时, 群内数据间距离就会突然增大, 对于群大于3时, 群内距离都比较小. 所以分为3群是比较好的选择.

plot(hclust.result, labels=iris$Species[index])

# 在每个聚类的群上添加外框线. k=3即在群数为3的情况下添加边框线

rect.hclust(hclust.result, k=3, border="red")

# 对数据进行分组. 显示出每一个数据对应的群组

groups <- cutree(hclust.result, k=3)

groups

94 129 95 2 34 97 75 100 78 40 130 41 116 141 37 26 32 43 137

1 2 1 3 3 1 2 1 2 3 2 3 2 2 3 3 3 3 2

21 6 29 104 67 138 128 140 57 33 136 61 22 90 24 31 115 93 63

3 3 3 2 1 2 2 2 2 3 2 1 3 1 3 3 2 1 1

73 35

2 3

# 与原数据中的分类方法进行对比

table(iris$Species[index], groups)

groups

1 2 3

setosa 0 0 16

versicolor 9 4 0

virginica 0 11 0

# 群级数的确定

# fpc包可以用来确定群级数,

library(fpc)

# 把iris数据集中是数字型的字段保留下来, 不是数值型的就跳过.

iris2 <- iris[,sapply(iris,is.numeric)]

iris2 <- iris[,-5]

iris2

# 把iris这个数据集分成了2群. 这个数据集有1群与其它群分开的比较多, 另外2群有重叠部分, 所以分成2群或3群都是可以接受的.

pamk.result <- pamk(iris2)

pamk.result$nc

# [1] 2

# NbClust可以用来确定群级数

library(NbClust)

# NbClust比pamk方法使用起来更灵活, 可以设定距离, 并且可以设置尝试分群的最小和最大群组数. 可以自定义方法, index="all"表示把NbClust分析的所有指标都输出来

NbClust.result <- NbClust(data = iris2, distance = "euclidean",

min.nc = 2, max.nc = 10,

method = "ward.D2", index = "all")

\*\*\* : The Hubert index is a graphical method of determining the number of clusters.

In the plot of Hubert index, we seek a significant knee that corresponds to a

significant increase of the value of the measure i.e the significant peak in Hubert

index second differences plot.

\*\*\* : The D index is a graphical method of determining the number of clusters.

In the plot of D index, we seek a significant knee (the significant peak in Dindex

second differences plot) that corresponds to a significant increase of the value of

the measure.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Among all indices:

\* 9 proposed 2 as the best number of clusters

\* 10 proposed 3 as the best number of clusters

\* 3 proposed 6 as the best number of clusters

\* 1 proposed 9 as the best number of clusters

\* 1 proposed 10 as the best number of clusters

\*\*\*\*\* Conclusion \*\*\*\*\*

\* According to the majority rule, the best number of clusters is 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 通过计算, 可以确定最佳的群级数为3. 从阶层式的聚类方法中也可以看出, 在群组数从2到3变化时, 距离会突然间变大.

NbClust.result$Best.nc

#### 阶层法对收入-年龄进行聚类分析

# 银行客户聚类分析. Age\_Income是一个人造数据, 每10笔数据分为一群.

age\_income <- read.csv("D:/R\_edu/data/Age\_Income.csv", header=T, sep=",")

# 读取ID部分, ID部分一般不会用来做聚类分析

age\_income\_ID <- age\_income[,1]

age\_income\_ID

# 做聚类分析时把第1个字段和最后一个字段都去掉, 只有年龄和收入两个字段

age\_income\_s <- age\_income[,-c(1,4)]

age\_income\_s

# 使用阶层式聚类方法

hclust.result <- hclust(dist(age\_income\_s), method= "ward.D2")

hclust.result

# 做图显示聚类的结果. lables使用原数据的ID. 聚类从3群变为2群时, 群内的距离就会变的非常大. 所以3群是最好的聚类结果

plot(hclust.result, labels=age\_income\_ID)

# 显示分群框

rect.hclust(hclust.result, k=3, border="red")

# 提取分组的结果

groups <- cutree(hclust.result, k=3)

# 以groups为颜色对原数据画图.使用阶层法时是没有群中心的, 只有kmeans方法才有群中心

plot(age\_income\_s, col=groups)

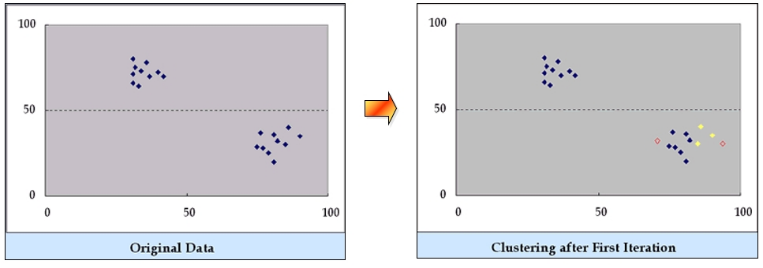
## 分割式聚类法:

一般情况下, 分割式聚类法的效果会比阶层式的效果好

### K均值法(K-Means Method)

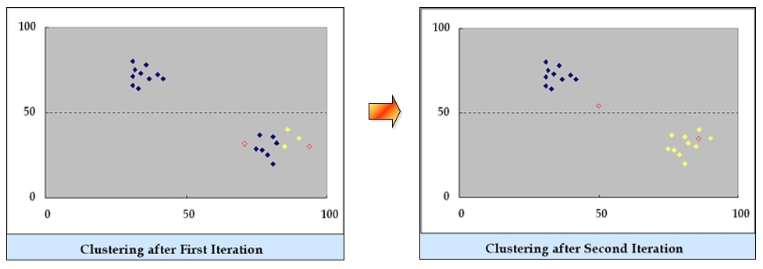
有20个数据. 输入变量有2个, 可以认为是x和y. 把20个数据作图, 得到散布图, 由散布图可以得知, 需要把数据分为2群.

第一步: K均值法会首先随机设定群中心, 群中心不是真实的数据点, 而是虚拟的点, 是由随机产生的数据生成的群中心. 确定了两个群中心之后, 画两个群中心的中线, 中线两边的数据距离中线两边的群中心的距离比较近. 只要群中心确定了, 每群中的数据也就确定了, 中线左边有17个数据点, 右边有3个数据点.

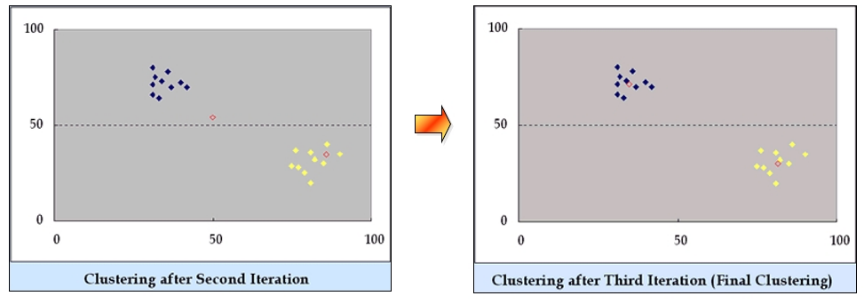


第二步: 重新计算群中心, 每个群中都以本群中所有数据重新计算群中心, 如以3个在中线右边的数据来计算这3个数据的群中心. 左边17个数据计算得到的群中心也会向左上方移动.

第三步: 得到两个新的群中心后, 再次作这两个群中心的中线, 再次以中线对数据进行分割. 中线左边群数据个数变为10个, 右边变为10个.



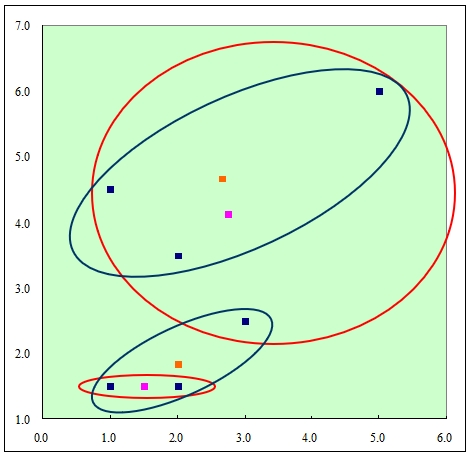
第四步: 再次计算2个群中的群中心, 然后作2个群中心的中线, 发现2个群中的数据并没有发生变化, 再次计算群中心的位置时群中心也不会发生变化, 只要群中心不发生变化, 分割式聚类的过程就完成了.



#### K均值法的优缺点

为什么称为K-Means? 这里的K表示要将数据分为K个群. 因为要多次计算群中心, 这里的Mean就是各个输入属性的平均值.

* Advantage
* Fast, 优势是速度快
* Disadvantages
* Sensitive to Outliers. 对离群值非常敏感. 如上面的数据, 有20个数据分成2群, 另外2个很遥远的位置处有2个离群值, 如果不进行处理, 得到的结果是2个离群值分为一组, 而20个数据分为一组. 所以使用这种方法时要特别注意离群值的处理, 一般是将离群值剔除掉. 如果聚类后发现某个群中数据个数特别少, 就可以认为是离群值, 就可以把它移除掉. 这个缺点就变成了识别离群值的一个比较好的方法.
* Not an Optimal Solution. 不是最佳的解决方案. 在刚开始时需要随机的产生群中心, 起始群中心产生的不同, 最终得到的聚类结果可能也是不同的. 所以不是最佳的解决方案, 最理想的情况应该是不论初始条件如何, 都能最终得到相同的结果. 如下图所示. 所以一般在使用K均值法时, 都会设置随机数种子, 一旦设定了随机数种子, 初始群中心的位置每次都是固定的了. 但这也并不是一个严重的问题, 如下图中, 有争议的只有一个数据点, 其它大部分数据的分类都是固定的, 不确定的这个数据点处在数据的中间位置, 所以可以用不同的方法划分为不同的群中.
* Can just handle numerical data. 只能处理数值型数据, 不能处理分类变量. 但这个问题也不严重, 因为分类型数据已经是分类的了, 使用分类型数据进行聚类本身的意义并不大. 如果非要考虑分类型变量, 如性别可以取男女2个值, 职业可以取市农工商4个值, 可以直接手动划分为8类, 然后分别对这8类进行聚类分析即可.



#### K均值法种子点的选取

* 种子点的选取关系着分类的结果。如果种子点选择不当，分类的结果就可能不甚理想
* 常用的方法有以下
* 随机选择K个没有缺失值的观察体, 如果程序包能处理缺失值, 就像上面计算距离矩阵时的方法, 就可以忽略, 如果程度包不能处理缺失值, 就要手动把缺失值移除或替换.
* 先选择第1 个观察体当第1 群的种子；其次选择与第1 个种子的距离超过既定标准的下一个观察体当第2 群的种子；接着选择与第1、2 个种子的距离超过既定标准的下一个观察体当第3 群的种子；依此类推，直到选出K个群的种子为止. 即尽量把群中心分的越散越好. 这个方法也仅作为参数.

#### 软件实现

##### k-Means对iris进行聚类分析

# 提出所有数值型变量

# Extract All Numerical Attributes

# 把iris数据集中是数字型的字段保留下来, 不是数值型的就跳过.

iris2 <- iris[,sapply(iris,is.numeric)]

iris2 <- iris[,-5]

iris2

# 使用kmeans有初始群中心的问题, 所以要先设置随机数种子

set.seed(1234)

# 把iris2数据集根据输入字段分为3群

kmeans.result <- kmeans(iris2, 3)

kmeans.result

K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 5.006000 3.428000 1.462000 0.246000

2 5.901613 2.748387 4.393548 1.433871

3 6.850000 3.073684 5.742105 2.071053

# 可以知道每个数据是被分到了哪个群中, 前中后50群分别分为1,2,3群.

Clustering vector:

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[71] 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3

[106] 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3

[141] 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:

[1] 15.15100 39.82097 23.87947

(between\_SS / total\_SS = 88.4 %)

Available components:

[1] "cluster" "centers" "totss" "withinss"

[5] "tot.withinss" "betweenss" "size" "iter"

[9] "ifault"

# 抽取出聚类的结果

kmeans.result$cluster

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[71] 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3

[106] 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3

[141] 3 3 2 3 3 3 2 3 3 2

# 查看原始分类与kmeans聚类分析得到的结果的差异

table(iris$Species, kmeans.result$cluster)

1 2 3

setosa 50 0 0

versicolor 0 48 2

virginica 0 14 36

# iris2中有4个变量, 就使用两两组合的方式把数据画出来. 颜色就使用聚类的结果,三个群级分别使用不同的颜色

plot(iris2, col=kmeans.result$cluster)

#Find 10 Largest Distances between Objects and Cluster Centers

# 找到10个群组中距离群组中心距离最远的对象. Kmeans方法是具有排它性的, 每一个数据只能属于一个群组. 但有些数据离群中心的实际距离也比较远, 它们只是相对来说离群中心比较近.

# 提取出群级的中心

kmeans.result$centers

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 5.006000 3.428000 1.462000 0.246000

2 5.901613 2.748387 4.393548 1.433871

3 6.850000 3.073684 5.742105 2.071053

# $cluster为分群情况,共150个数据, $center是群中心, 得到150个数据, 如果是第1群的, 就把第1群的群中心信息贴到后面, 如果是第2群的, 就把第2群的群中心贴到后面

centers <- kmeans.result$centers[kmeans.result$cluster,]

nrow(centers)

[1] 150

head(centers, 3)

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 5.006 3.428 1.462 0.246

1 5.006 3.428 1.462 0.246

1 5.006 3.428 1.462 0.246

tail(centers, 3)

Sepal.Length Sepal.Width Petal.Length Petal.Width

3 6.850000 3.073684 5.742105 2.071053

3 6.850000 3.073684 5.742105 2.071053

2 5.901613 2.748387 4.393548 1.433871

# iris2中是每个数据的位置, centers是每个数据对应的群中心的位置, 把它们相减,得到每个数据到群中心的距离. 距离的平方和就是误差的平方和. 开根号就得到每个数据距离自己所在群中心的欧几里德距离.

distances <- sqrt(rowSums((iris2 - centers)^2))

distances

# 通过order函数对距离进行排序,找到最大的10个数, 返回索引值

outliers <- order(distances, decreasing=T)[1:10]

outliers

[1] 99 58 94 61 119 118 132 123 42 51

# 从iris2中取出来这10个到组中心距离最大的10个数

iris2[outliers,]

Sepal.Length Sepal.Width Petal.Length Petal.Width

99 5.1 2.5 3.0 1.1

58 4.9 2.4 3.3 1.0

94 5.0 2.3 3.3 1.0

61 5.0 2.0 3.5 1.0

119 7.7 2.6 6.9 2.3

118 7.7 3.8 6.7 2.2

132 7.9 3.8 6.4 2.0

123 7.7 2.8 6.7 2.0

42 4.5 2.3 1.3 0.3

51 7.0 3.2 4.7 1.4

# 画图, 把Sepal.Length和Petal.Width画图

plot(iris2[c("Sepal.Length", "Petal.Width")], col=kmeans.result$cluster)

# 在图上添加群中心, 只提取出Sepal.Length和Petal.Width的数据. col为颜色, 3个群组使用不同的颜色. pch=8自定义符号为\*号, cex=2表示\*号的大小为默认值的2倍

points(kmeans.result$centers[,c("Sepal.Length", "Petal.Width")], col=1:3, pch=8, cex=2)

# 把离群值添加到图上.

points(iris2[outliers,c("Sepal.Length", "Petal.Width")], col=4, pch='+', cex=2)

##### K-Means对收入-年龄进行聚类分析

# 使用K-Means法进行聚类

# K-Means Clustering

age\_income <- read.csv("D:/R\_edu/data/Age\_Income.csv", header=T, sep=",")

age\_income\_s <- age\_income[,-c(1,4)]

# 使用kmeans来做聚类分析, 群级数定为3

kmd <- kmeans(age\_income\_s, centers=3)

kmd

-means clustering with 3 clusters of sizes 10, 10, 10

Cluster means:

Age Income

1 22.7 85

2 38.8 45

3 56.0 97

Clustering vector:

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:

[1] 956.1 895.6 1540.0

(between\_SS / total\_SS = 85.7 %)

Available components:

[1] "cluster" "centers" "totss" "withinss"

[5] "tot.withinss" "betweenss" "size" "iter"

[9] "ifault"

# cluster中显示30个数据中每个数据都属于哪一群.

kmd$cluster

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

# 聚成3群, 显示每一群的群中心, 即年龄和收入的平均值, 群中心也可以在一定程度代表这一群, 所以可以使用群中心对这一群进行描述

kmd$centers

Age Income

1 22.7 85 富二代

2 38.8 45 年老多金型

3 56.0 97 年老贫穷型

# Draw Graph 画图, 颜色为对应的群号

plot(age\_income\_s, col=kmd$cluster)

# 标出群中心的位置

points(kmd$centers, col=1:3, pch=8, cex=2)

# Clustering Evaluation 对群级进行评估, 看分成几个群级效果最好

# R2

# $totss是当聚为一群时的SS值, 即每一点到群中心位置的平方和

kmd$totss

[1] 23764.83

# $withinss为每一群对应的SS值

kmd$withinss

[1] 956.1 895.6 1540.0

# 分为三群时总的ss值,等于每一群的ss值相加

kmd$tot.withinss

[1] 3391.7

# 计算R^2, 三群总的ss值越小, R2的值越大, 所以R2的值越大越好. 在聚类得到的群数逐渐减小时, 4->3->2->1的过程中, R2是逐渐减小的, 但不能突然间减小.

R2 <- (kmd$totss - kmd$tot.withinss) / kmd$totss

R2

[1] 0.8572807

# kmd$betweenss=kmd$totss-kmd$tot.withinss

R2 <- kmd$betweenss / kmd$totss

R2

[1] 0.8572807

# Test R2 for Different Different K

# 随着分群数k的变化R2的变量情况.

result <- list()

# 从2群到5群时, R2的变化情况

for (i in 2:5){

kmd <- kmeans(age\_income\_s, centers=i)

R2 <- kmd$betweenss / kmd$totss

result[[paste('k=',i,sep='')]] = R2

}

# 在群数从5减小到2的过程中, R2是逐渐减小的, 但在从3群变为2群时, R2有一个突然的变化. 所以3群时是一个最好的结果

result

$`k=2`

[1] 0.5938984

$`k=3`

[1] 0.8572807

$`k=4`

[1] 0.8796674

$`k=5`

[1] 0.9135726

# 使用轮廓系数s(i)来衡量分群的好坏. 轮廓系数是用每一个点与它所在群的其它点的平均距离a(i)和这个点与它最近邻的群中每个点的平均距离b(i)的比较来衡量分群效果的好坏. 当一个点到它所在群的其它点的平均距离a(i)越小, 而到与它最近邻的群中每个点的平均距离b(i)越大时, 分群的效果越好. 也就是说轮廓系数越大时效果越好. 当分为N个群时, 每个群可以计算出一个轮廓系数, 总体也可以计算出一个轮廓系数. 总体的轮廓系数越大分群的效果越好. 同时还要考虑数据在各个群中的分布是否均匀.

# 使用cluster包来计算轮廓系数

# Test Silhouette Coefficient for Different Different K

library(cluster)

age\_income <- read.csv("D:/R\_edu/data/Age\_Income.csv", header=T, sep=",")

age\_income\_s <- age\_income[,-c(1,4)]

# 使用kmeans进行聚类

kmd <- kmeans(age\_income\_s, centers=3)

?silhouette

silhouette(x, dist, dmatrix, ...)

# 计算轮廓系数, 给出聚类的结果, 即对应的数据是属于哪一群的, 再给定两两一对的距离矩阵.

sil <- silhouette(kmd$cluster, dist(age\_income\_s))

sil

# cluster显示出30个数据属于分群后属于哪个群体, neighbor为与它最近邻的群组是哪个. sil\_width是计算出来的每个数据对应的轮廓系数.

cluster neighbor sil\_width

[1,] 2 1 0.4494290

[2,] 2 3 0.6938641

[3,] 2 1 0.7336477

[4,] 2 3 0.5145561

[5,] 2 1 0.5983362

[6,] 2 3 0.5751620

[7,] 2 3 0.6667986

[8,] 2 3 0.6093908

[9,] 2 3 0.6805783

[10,] 2 1 0.7336477

[11,] 1 2 0.6732389

[12,] 1 2 0.3601289

[13,] 1 2 0.7424495

[14,] 1 2 0.6201774

[15,] 1 2 0.7949620

[16,] 1 2 0.7421761

[17,] 1 2 0.7829122

[18,] 1 2 0.7730479

[19,] 1 2 0.7259441

[20,] 1 2 0.7936143

[21,] 3 2 0.1572720

[22,] 3 2 0.5252222

[23,] 3 2 0.4326285

[24,] 3 2 0.6270225

[25,] 3 2 0.4498124

[26,] 3 2 0.6329920

[27,] 3 2 0.6914412

[28,] 3 2 0.6018783

[29,] 3 2 0.6820390

[30,] 3 2 0.6692599

attr(,"Ordered")

[1] FALSE

attr(,"call")

silhouette.default(x = kmd$cluster, dist = dist(age\_income\_s))

attr(,"class")

[1] "silhouette"

# 把所有数据的轮廓系数求平均, 就得到了分为三群时总体的轮廓系数, 即ppt中轮廓系数的虚线部分.

mean(sil[,'sil\_width'])

[1] 0.6244543

result <- list()

# 计算分群数为2:5时总体的轮廓系数.

for (i in 2:5){

kmd <- kmeans(age\_income\_s, centers=i)

sil <- silhouette(kmd$cluster, dist(age\_income\_s))

result[[paste('k=',i,sep='')]] <- mean(sil[,'sil\_width'])

}

# 从结果中可以看出, 当k=3时轮廓系数是最大的. 也刚好把所有的数据分为三群, 每群中有10个数据, 即是最平均的分配方法. 所以k=3是最佳的分配方式.

result

$`k=2`

[1] 0.5206906

$`k=3`

[1] 0.6244543

$`k=4`

[1] 0.5822845

$`k=5`

[1] 0.493592

### PAM算法原理

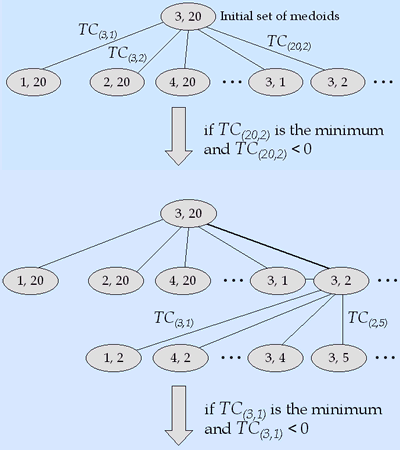
与K均值法比较类似, 但是它不会随机选取虚拟的群中心, 而是每次随机选取真实的数据点作为群中心, 每一步都保证得到的结果是最优的结果. 使用真实数据点作为群中心, 就可以解决很多问题, 如K均值法不能处理类别型变量的问题.

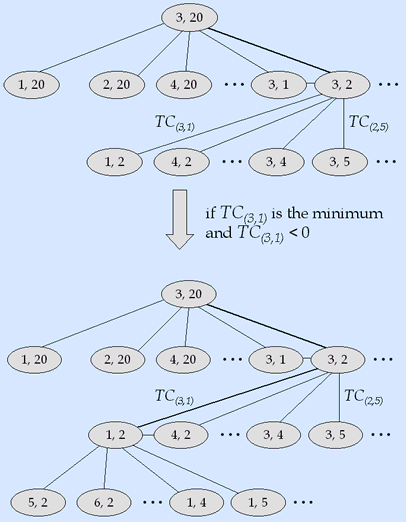
如下面的例子, 先随机挑选2个数据点, 3, 20作为2个群中心, 然后计算SS值. 然后分别交换3和20这两个值, 把3换为其它的值, 即选择(1,20), (2,20), (4,20), …, (19,20)作为群中心, 计算SS的值, 再把20换为其它的值, 即选择(3,1), (3,2), …, (3,19)作为群中心, 计算SS的值. 从包括(3,20)组合在内的所有的SS值中选出最小的那一个, 它对应的群中心就是要进行分群的群中心.

例如找到了(3,2)这个群中心在所有的组合中SS值是最小的. 然后再次对(3,2)这个组合中的2个数进行替换, 计算SS值, 包括(3,2)在内的SS值最小的组合就是要求的群中心.

假如第二步中找到的最小值的组合是(1,2), 再使用上面的方法进行替换并比较SS值, 如此一直进行下去, 直到找不到更小的组合为止.

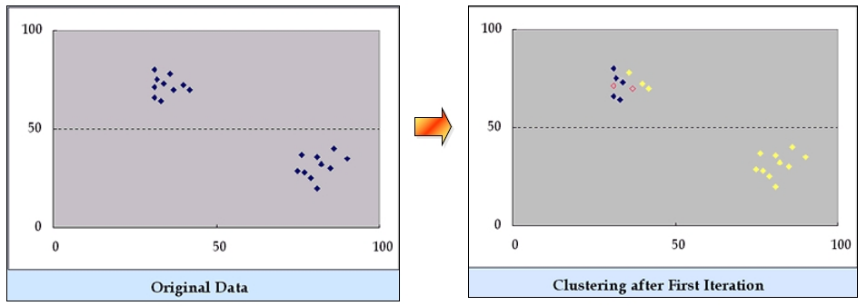
PAM的基本思想就是使用真实的数据点对群中心进行替换, 找到SS值最小的组合. 这样就可以处理类别型字段. 并且PAM方法能够保证找到最佳组合. 但PAM算法非常非常的耗时, 特别是在数据量特别大时.



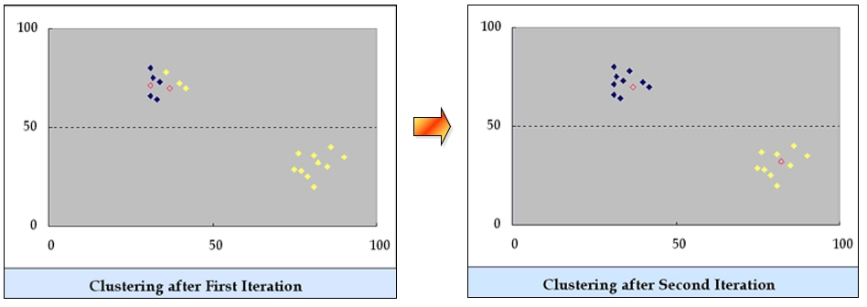


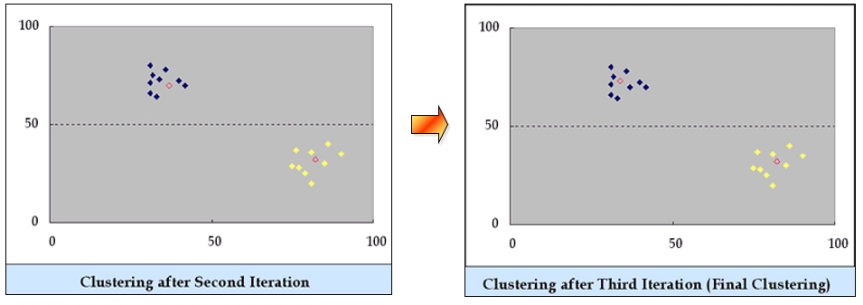
#### PAM算法分组图示

首先任意选取两个数据点作为群中心, 以其中线对数据进行分割



把所选群中心的一个点换成另一个数据点作为群中心. 计算SS值, 多次替换找到SS值最小的群中心组合, 再次对数据进行分割. 再次进行群中心的替换, 计算SS值, 取SS值最小的群中心组合, 如此进行下去, 直到找到SS值最小的那个组合.





#### K均值法和PAM算法的比较

* K-Means can handle large data sets efficiently but is limited to interval-scaled attributes and sensitive to outlier. K均值法可以有效的处理大的数据集, 但是只能处理数值型的变量, 并且对离群值非常敏感. 如果有分类型的变量, 必须要手动进行处理.
* PAM is capable of handling various attribute types but is not efficient when clustering large data sets. PAM法可以处理多种数据类型, 对离群值不敏感, 但处理大的数据集时效率很低.

#### 软件实现

##### PAM法对收入-年龄进行聚类分析

# 使用PAM聚类方法自动获取K值. PAM与kmeans方法的不同之处在于使用真实的数据点作为群中心

# Autumatically Finding K Using PAM

library(fpc)

# 使用PAM方法时会自动使用轮廓系数决定群级数, 所以不用手动指定群组数.

pamk.result <- pamk(age\_income\_s)

pamk.result

$pamobject

Medoids:

ID Age Income

[1,] 7 28 85

[2,] 17 44 45

[3,] 27 59 95

Clustering vector:

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

Objective function:

build swap

10.030567 9.738534

Available components:

[1] "medoids" "id.med" "clustering" "objective" "isolation"

[6] "clusinfo" "silinfo" "diss" "call" "data"

$nc

[1] 3

$crit

[1] 0.0000000 0.5400414 0.6244543 0.5313622 0.4833998 0.4185080 0.4367567

[8] 0.4318734 0.4184464 0.3728518

# 查看分组的结果

pamk.result$pamobject$clustering

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

# 查看分群级数

pamk.result$nc

[1] 3

# Graph for PAM

# 作图, clusplot,为分群的结果, silhouette plot of pam显示每一群中每一个数据的轮廓系数, 并按从大到小的顺序排列

layout(matrix(c(1,2),1,2))

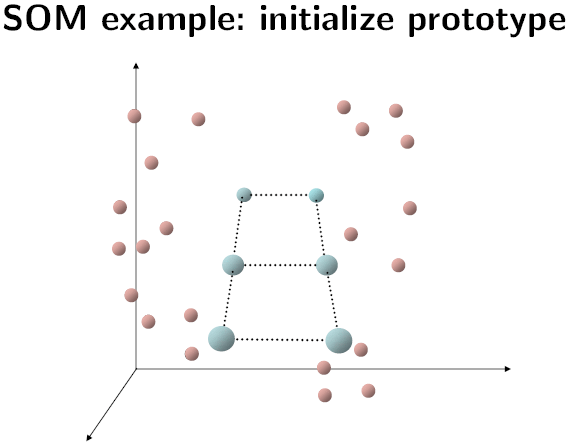
plot(pamk.result$pamobject)

layout(matrix(1))

## SOM(Self-organizing Maps)算法

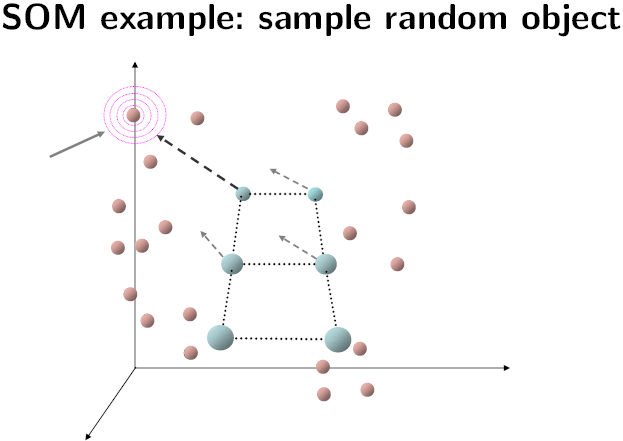
**神经网络的聚类方法. Kohonen所提出的算法**

### initialize prototype



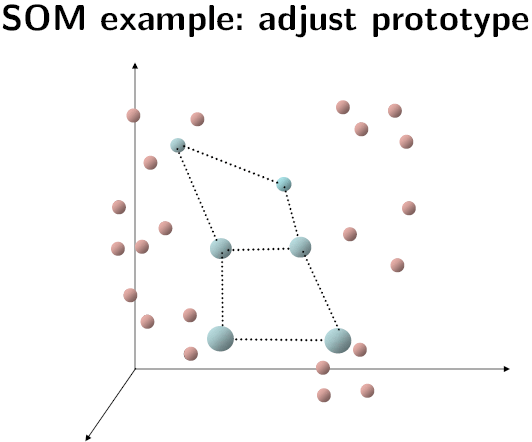
不足之处, 它的群级数是矩阵的情况下才能得到一个比较好的结果, 如(3\*2)矩阵的群级数, 即分成6群. 在类似(2\*2), (3\*2)的群级数时效果比较好, (1\*2), (1\*3)的群级数下效果不好. 这里数据看似比较符合6群, 所以使用(3\*2), 选择6个群中心点.

### sample random object

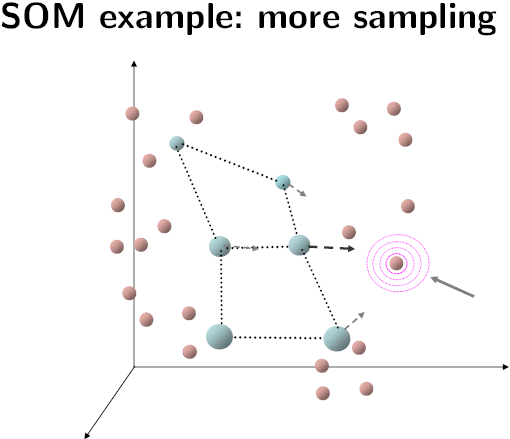


左上角第一个数据点进入模型之后, 优胜点是左上角的, 所有的群中心点都会向左上角移动, 优胜点的移动距离最大, 其它点随着离左上角的距离越远移动的距离越小.

### adjust prototype

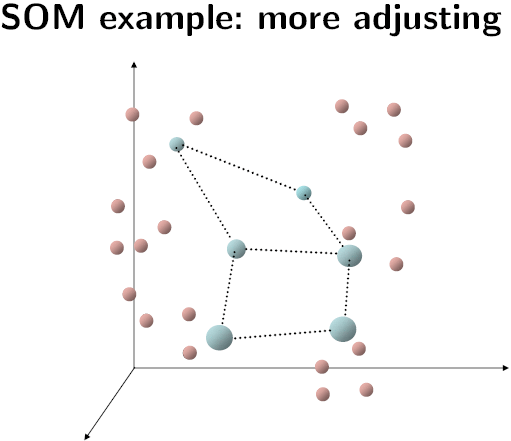


### more sampling



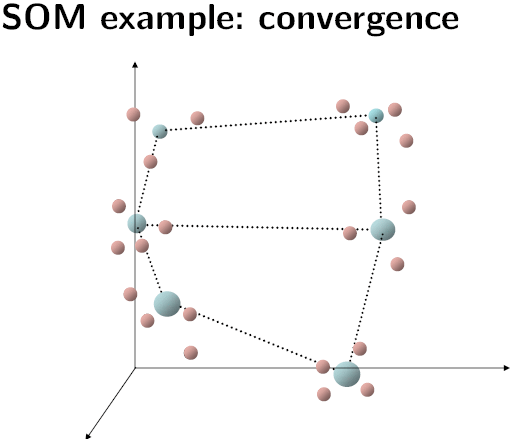
进入第2个数据点, 会把所有的群中心向这个数据点拉动, 优胜点拉动的距离最大.

### more ajusting



### convergence

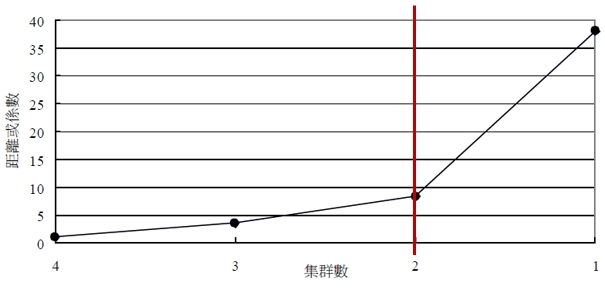
所有点都考虑进去之后, 还要继续运行, 直到达到收敛条件, 群中心不再发生变化时, 就达到了预想的条件. 这样, 每个群中心附近的数据点都可以分成一群.



## 群数的判断

### 陡坡图法

* 在分割式的聚类方法中, 必须要先确定群级数. 如使用K均值法时必须要先确定需要把数据聚为多少群.
* 如果使用阶层式聚类分析法，在集结完成后，接下来就要判断应分成几个群集才算恰当，一般可以用陡坡图来判断
* 之前使用华德法进行聚类分析的例子，各阶段都有SS系数的计算, SS系数就是用来判断聚类结果的好坏. 群级数越少, 群内的差异性就越来越大, SS系数也会越来越大. 各阶段的SS系数分别为1.00、3.50、8.33、38.00，在从4群到3群, 从3群到2群的过程中, SS系数可以视为是线性增加的, 但从2群到1群的过程中, SS值有一个很大的变化, 这说明把数据分为2群时是比较合理的, 此时SS值和群数达到了一个最优的组合, 在从2群变为1群时, 就打破了这种最优的组合, 造成了SS值的快速增大, 所以可以从SS值的变化来判断分群数是否合理, SS值由线性到快速增大时的点就是最佳群数的分隔点. SS值的陡坡图如下



* 有三种常用的群数判断方法
* R-Squared (R2)
* Semi-Partial R-Squared, 前两种方法是SS值判断方法的变形.
* Silhouette Coefficient (轮廓系数)

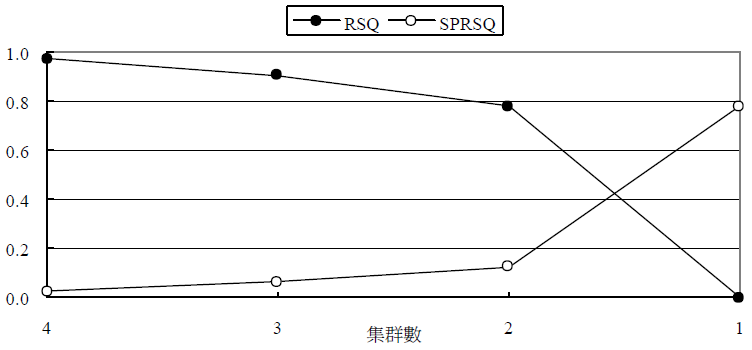
### 群数的判断(R2)

SS可分为组间 (SSB)、组内 (SSW)及全体 (SST)，SSB+SSW=SST

R2代表某个群数下的SS距离总的SS还有多远, R2的值越大, 离总的SS的距离越远.

* 1. 随着观察体的合并，集群数目愈来愈少，集群内的异质性愈来愈高，因此SSW就会愈来愈大.
  2. 由于聚类分析的主要目的是使集群内的变异尽可能变小，而使集群间的变异尽可能的大，因此R2 要大一些
  3. 如果由于观察体的合并，使得R2突然减小，表示应停止合并

上一部分我们用华德计算出集结的联合组内SS分别是1.00, 3.50, 8.33, 38.00, 而总SS是所有观察体合并时的38.00. 因此组间R2应该是 , , , . 从2群到1群时, R2由之前的线性缓慢减小变为突然间有一个大的下降, 此时就应该停止合并, 最佳的分隔群数为2.



### 群数的判断(Semi-Partial R2)

* 半净 R2(Semi-Partial R-Squared)，它代表由于加入新的观察体到集群中，比上一个阶段增加的异质性 (也就是减少的同构型)
* 半净R2要小一些，如果半净R2突然增加，就表示集结过程应该停止

5群时的SS值为0, 从5群到4群时, SS值增加了1.00, 1.00-0表示增加的SS值, 增加的SS值除以SS值的最大值38, 得到的系数就是半净R2. 在半净R2急剧增加时, 就应该停止合并分群增加群级数.

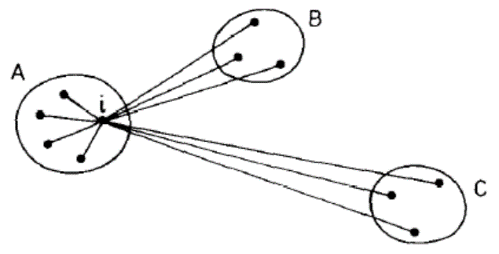
在本例中, 半净R2分别是, , ,

如果是阶层式的聚类方法, 也可以根据群数发生变化时SS值的变化来确定分群的最佳群级数.

如果是分隔式的聚类方法, 例如K-Means, 就只能进行尝试, 先把数据分为5群, 再4群, 3群, 最后1群, K-Means要做多次, 每一次都会得到一个SS值, 同样可以通过陡坡图, R2和semi-R2来判断最佳的群数.

### 轮廓系数(Silhouette Coefficient)

* a(*i*) is the average distance of *i* with all other data within the same cluster. We can interpret a(*i*) as how well *i* is assigned to its cluster
* b(i) is the lowest average distance of i to any other cluster, of which i is not a member. The cluster with this lowest average dissimilarity is said to be the “neighboring cluster” of i because it is the next best fit cluster for point i



每一个资料点都会计算出一个轮廓系数, *i*这个资料点被分到了A群中.

*a*(*i*) = average dissimilarity of *i* to all other objects of A.

*a*(*i*)就是i和同一群中所有资料点距离的平均值.

*d*(*i,C*)=average dissimilarity of *i* to all objects of C.

d(i,C)就是i资料点同C群中所有资料点的距离的平均值.

b(i)就是i资料点与其它最靠近i资料点所在群的群中资料点之间距离的平均值. 即分别计算i到其它群中所有资料点的平均距离, 取其中平均距离最小的值. 正常情况下, b(i)是小于a(i)的. 即i与同一群中的点越相似越好, 与不同群的差异性越大越好, 越不相似越好. a(i)的值一般比较小, b(i)的值一般比较大.

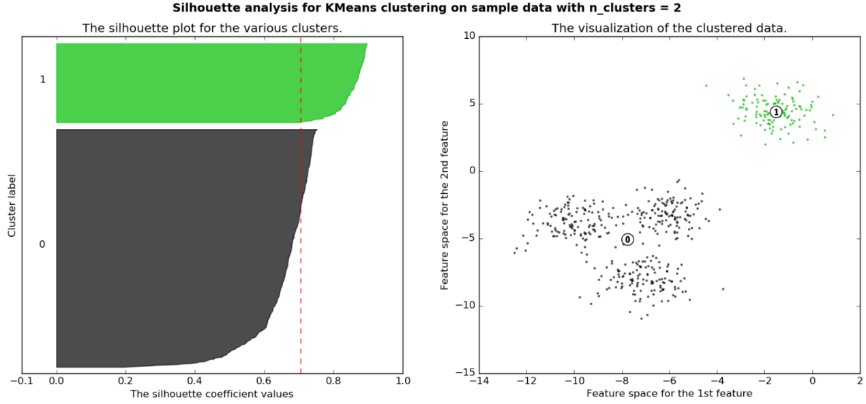
每一笔数据, 每一个资料点都会计算出一个轮廓系数, 要求平均的轮廓系数, 它的值越大越好.

b(i)一般会大于a(i), 不同群间的差异性越大, 或同一群内的相似性越大, s(i)就越大. 即轮廓系数越大的话, 分群的结果就越好.

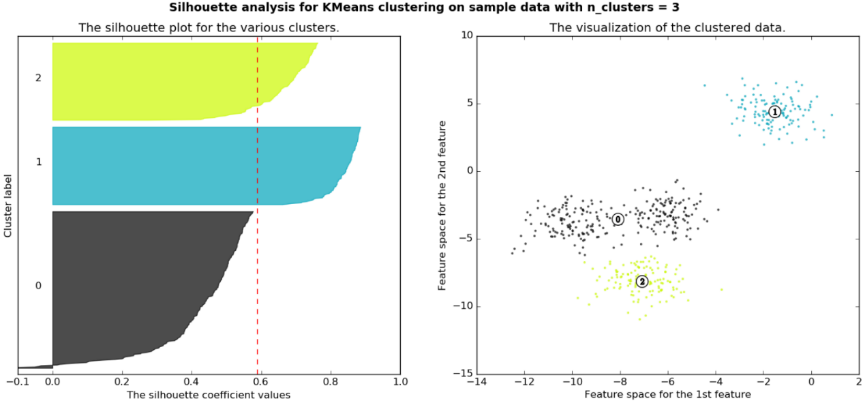
* The average s(*i*) over all data of the entire dataset is a measure of how appropriately the data have been clustered
* Silhouette plots may be used to determine the natural number of clusters within a dataset

### 整体平均轮廓系数

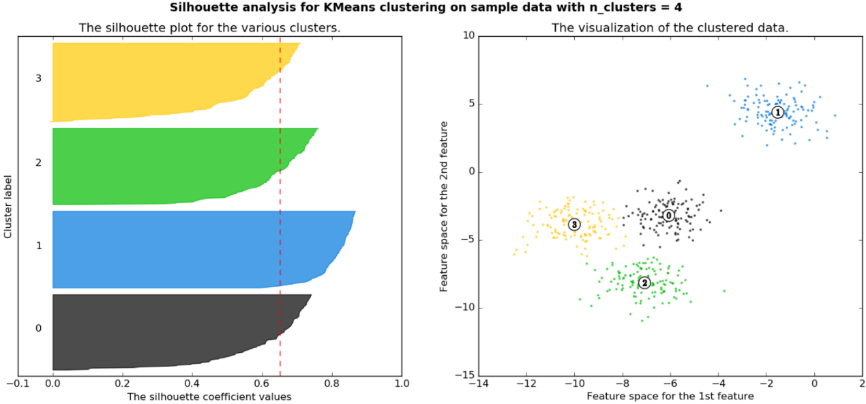
每一个资料点都会计算得到一个s(i)的值, 对应一个轮廓系数, 所有点的轮廓系数计算完后, 再计算整体的平均轮廓系数.



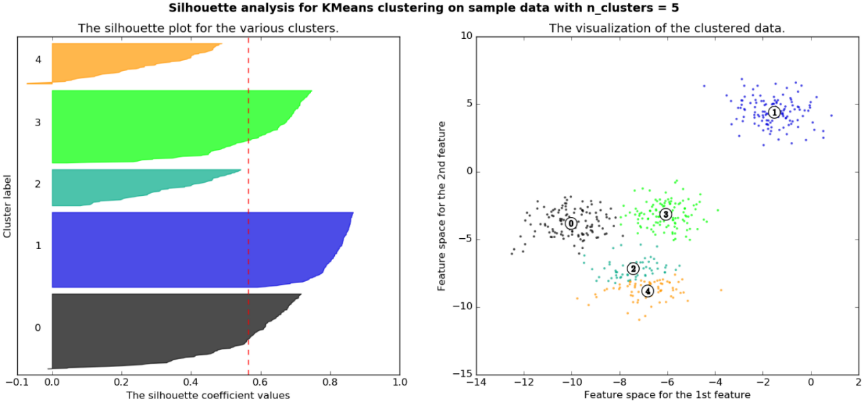
计算1群所有点的轮廓系数, 从大到小排列, 再计算0群中所有点的轮廓系数, 从大到小排列. 左图中的虚线就是所有轮廓系数的平均值. 平均轮廓系数就是用来评估聚类结果好坏的因素. 聚类分为2个群时平均轮廓系数约为0.7左右.



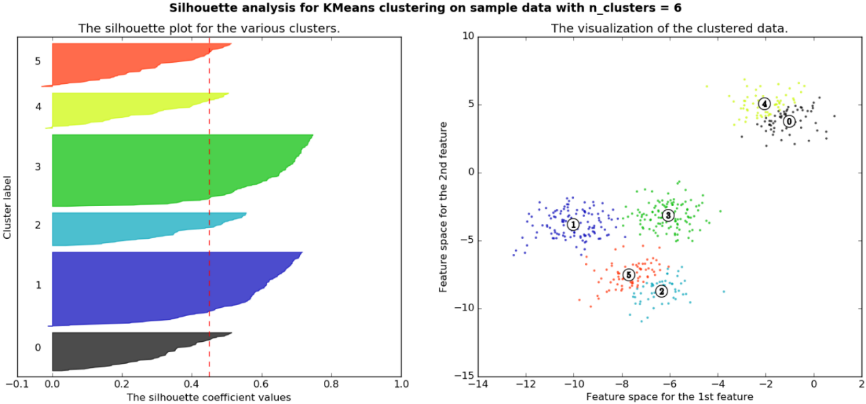
聚类分为3个群时平均轮廓系数约为0.6左右.



聚类分为4个群时平均轮廓系数稍微大于0.6, 在0.65左右.



聚类分为5个群时平均轮廓系数约为0.58左右.



聚类分为6个群时平均轮廓系数约为0.45左右.

* For n\_clusters = 2
* The average silhouette\_score is 0.704978749608
* For n\_clusters = 3
* The average silhouette\_score is 0.588200401213
* For n\_clusters = 4 (需考虑群大小是否分布平均)
* The average silhouette\_score is 0.650518663273
* For n\_clusters = 5
* The average silhouette\_score is 0.563764690262
* For n\_clusters = 6
* The average silhouette\_score is 0.450466629437

n=2时轮廓系数最大, n=4时轮廓系数为次最大. 如果完全按照轮廓系数进行判断, n=2时为最好的分群方法, 但此时两个群中数据分布比例不太匹配, 一个群中数据量较多, 而另一个群中数据量较少. 如果考虑到数据在各群中是否平均分布, 即每个群内数据分布的比例是否大致相同, n=4时是最佳的分群方法.

总结: 群数的判断, SS值是对阶层式聚类中华德法评估指标的延伸, SS值也是评估聚类效果好坏的一个指标. SS值搭配陡坡图就可以对群级数进行判断. 可以把SS值变形为R2, Semi-R2或者轮廓系数.

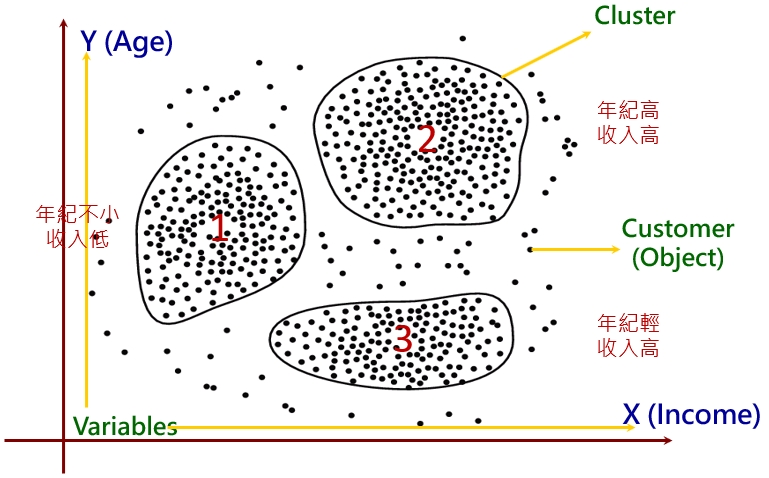
## 群体特征的描述

* 聚类分析所关心的重点有三
  1. 如何以数字来表示成员间的相似性 (或相异性)？
  2. 如何根据这些相似性将类似的成员分在同一群？
  3. **所有成员分群完毕后，对每一群的特征应如何描述？**

对群体特征的描述就要看输入字段, 从输入字段可以描述每个群体的特征. 如银行客户聚类中, 想要把收入和年龄相近的客户聚集到一起. 可以计算一下某个群体中所有客户年纪和收入的平均值, 再结合直方图看年龄和收入的分布情况, 就可以对这个群体进行描述, 如群体1的标签就是"年纪不小, 收入低".

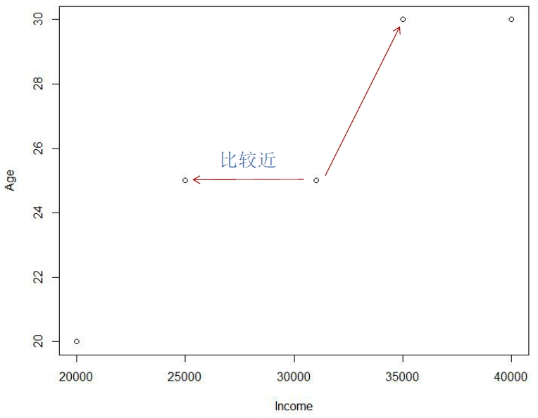
在聚类完成后, 就可以使用输入字段的平均值来描述每个群体与其它群体之间的差异性. 所以在这个例子中, 输入字段是年龄和收入, 就可以使用平均年龄和平均收入对各个群体进行描述.

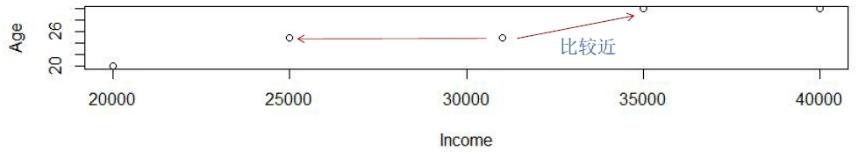
### 聚类分析的应用：银行客户聚类



## 聚类的正规化问题

在之间的聚类操作中, 只要是数值型的数据, 都用来直接进行聚类分析了, 这样做是有一定的风险的, 必须要考虑聚类的正规化, 标准化的问题.





以Age-Income为例, 假如有5个资料点, 想要把数据聚成2群, 最左边2个比较接近, 最右边2个也比较接近, 只有中间右边的数据不能确定是分到哪一群中. 从第一个图中觉得它与左边两个数据点比较接近, 但是如果把Age坐标轴缩的很小, 而Income坐标辆拉伸的很大, 这时中间右边的数据点就在视觉上与右上角的群比较接近了. 所以要把它与右边的两个数据聚成一群. 所以就有着数据的标准化的问题. Income的差距都在几千几万, 而Age的差距在几十之间, 如果两个点Age的差距为10, Income的差距为5000, 总的差距为5010, 这样计算, 因为数据本身的量级的差距, 就把年龄的重要性降低了, 实际上是主要是使用Income的值来进行聚类的.

应该要进行数据正规化‚将数据正规化成0~1之间‚年龄和收入的最大值都正规化为1, 最小值都正规化为0, 这样年龄和收入的地位就一样了, 不会因为数据本身量级的问题造成聚类进偏重使用某一个数据. 图1中所做的图就是已经标准化后的数据. 在这个图中, 可能10岁的差距对年龄来说就是很大的差距了, 而就算是5000的收入差距可能也不是太明显.

### 软件实现

#### 聚类数据的正规化/标准化

#Normalization for Clustering

age\_income <- read.csv("D:/R\_edu/data/Age\_Income4.csv",header=T,sep=",")

plot(age\_income)

set.seed(105)

# 使用kmeans方法把数据聚成2群

kmd <- kmeans(age\_income, centers=2)

# 查看聚类的中心

kmd$centers

Income Age

1 22500.00 22.50000

2 35333.33 28.33333

# 以不同的颜色进行聚类, 可以看到中间右边的数据与右上角的两个数据聚成了一群.

plot(age\_income, col=kmd$cluster)

# 添加聚类的中心

points(kmd$centers, col=1:3, pch=8, cex=2)

# 使用极值正规化来对数据进行正规化. 把最大值标准化为1, 最小值标准化为0, 其它数据都处于0和1之间

# Normalization, 定义一个标准化函数

normalize <- function(x) {

return((x - min(x)) / (max(x) - min(x)))

}

# 对数据进行标准化

age\_income\_sn <- as.data.frame(lapply(age\_income, normalize))

age\_income\_sn

Income Age

1 0.00 0.0

2 0.25 0.5

3 0.55 0.5

4 0.75 1.0

5 1.00 1.0

plot(age\_income\_sn)

# 重新进行kmeans标准化

set.seed(105)

kmd <- kmeans(age\_income\_sn, centers=2)

# 查看聚类的中心

kmd$centers

Income Age

1 0.2666667 0.3333333

2 0.8750000 1.0000000

# 作图, 现在就可以看到, 中间右边的数据与左下角两个数据聚为了一群. 所以在做聚类之前一般都要进行标准化的操作, 才能得到正确的结果.

plot(age\_income\_sn, col=kmd$cluster)

points(kmd$centers, col=1:3, pch=8, cex=2)

## R语言实操

* 以R语言进行分割式聚类分析及算法参数调整
* 以R语言进行阶层式聚类分析及算法参数调整

### 银行客户聚类实作

* 一家银行希望使用客户的相关讯息来将全行客户分群，以利后续相关营销活动的规划及进行
* 该数据保存在banking\_cust这张表中；其中CID为顾客身分ID，不作为聚类所用的输入字段； Sex及Marriage也仅仅作为参考字段，不作为聚类所用的输入字段；除此之外的其它变量(Income及Age)可以用于构造聚类模型
* 具体的变量描述如下：

主要还是使用Income和Age来进行聚类

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 字段说明 | 字段角色 |
| CID | 数值 | 客户代码 | 身份ID |
| Income | 数值 | 客户年收入 | 输入字段(Input Attribute) |
| Sex | 字符 | 性别 | 参考字段(Reference Attribute) |
| Age | 数值 | 客户年龄 | 输入字段(Input Attribute) |
| Marriage | 字符 | 客户婚姻状态 | 参考字段(Reference Attribute) |

* 假设银行想利用K-means的算法来将客户分成3群。请使用R软件完成以下题目：

1、(A)

* + 1. 数据集的大小(笔数)为何?
    2. A.30 B.31 C.29 D.28

2、(B)

* + 1. Income的标准差为多少?
    2. A.44.231 B.46.294 C.33.245 D.55.121

3、(B)

用K-means聚类后，笔数最少的群有多少笔数据?

A.10 B.2 C.8 D.7

* + 1. 请移除笔数最少的群的数据，然后再利用K-means(K设为3)进行聚类一次。输入变量一样使用Income及Age。
    2. 4、(A)
    3. 移除笔数最少的群的数据后，Income的标准差为多少?
    4. A.25.016 B.33.271 C.52.016 D.46.251

5、(C)

再次进行聚类聚类后，笔数最少的群有多少笔数据?

A.10 B.4 C.2 D.7

* 1. 请再次移除笔数最少的群的数据，然后再利用K-means (K设为3)进行聚类一次。输入变量一样使用Income及Age。
  2. 6、(C)
  3. 移除数据前后，Age的标准差的差值为多少?
  4. A.15.297 B.8.412 C.5.297 D.4.143

7、(C)

进行聚类聚类后，3群分别有多少笔数据?

A.10/10/10 B.9/8/7 C.8/10/8 D.7/10/6

* + 1. 8、(A)
    2. 最后过滤掉多少笔离群值的数据?
    3. A.4 B.3 C.2 D.5

9、(A)

笔数最多的那一群，其特征描述为何?

A.年纪不轻，收入少 B.年纪轻，收入高

C.年纪高，收入高 D.年纪高，收入少

* 1. 10、(B)
  2. 笔数最多的那一群，其男生的比例为何?
  3. A.60% B.80% C.50% D.70%

#### K-Means聚类方法实现

# Example for K-Means Clustering

age\_income <- read.csv("D:/R\_edu/data/Age\_Income3.csv",header=T,sep=",")

# 主要以Income和Age进行聚类, 所以把1,3,5去除掉

age\_income\_s <- age\_income[,-c(1,3,5)]

# 查看有多少笔资料

nrow(age\_income\_s)

[1] 30

# 计算标准差

sd(age\_income\_s$Income, na.rm = TRUE)

[1] 46.29367

# 原数据的分布, 从中可以看到, 年龄和收入都有离群值.

plot(age\_income\_s)

# 定义标准化函数

normalize <- function(x) {

return((x - min(x)) / (max(x) - min(x)))

}

# 对数据进行标准化

age\_income\_sn <- as.data.frame(lapply(age\_income\_s, normalize))

# 数据的整体分布没有发生变化, 坐标轴的范围发生了变化

plot(age\_income\_sn)

set.seed(100)

# 使用kmeans进行聚类分析, 聚为3群.

kmd <- kmeans(age\_income\_sn, centers=3)

# 把两个Income的离群值聚为了一群.

plot(age\_income\_sn, col=kmd$cluster)

# 在图上添加群中心点

points(kmd$centers, col=1:3, pch=8, cex=2)

# 列联表分析聚类的结果, 显示每一群中的数据个数. 如果某一群中数据个数特别少时, 一般就认为这是离群值

table(kmd$cluster)

1 2 3

2 19 9

kmd$cluster

# 如果存在着离群值, 就要把离群值移除再进行聚类. 群1是离群值, 所以把群1从数据中移除

age\_income\_rm <- age\_income\_s[kmd$cluster!=1,]

# 进行标准化

age\_income\_rmn <- age\_income\_sn[kmd$cluster!=1,]

# 查看标准化后的数据分布

plot(age\_income\_rmn)

# 计算移除离群值后年龄和收入的标准差

sd(age\_income\_rm$Income, na.rm = TRUE)

sd(age\_income\_rm$Age, na.rm = TRUE)

# 重新进行聚类分析

set.seed(201)

kmd <- kmeans(age\_income\_rmn, centers=3)

# 查看聚类后的群中心

kmd$centers

Income Age

1 0.3333333 0.9562500

2 0.1534392 0.2104167

3 0.3154762 0.5093750

# 作图查看聚类的结果. 又把年龄的两个离群值分为了一类

plot(age\_income\_rmn, col=kmd$cluster)

points(kmd$centers, col=1:3, pch=8, cex=2)

# 查看聚类的结果

kmd$cluster

# 使用列联表对聚类结果进行分析

table(kmd$cluster)

1 2 3

2 18 8

# 再次移除离群值并进行标准化

age\_income\_rm2 <- age\_income\_rm[kmd$cluster!=1,]

age\_income\_rm2n <- age\_income\_rmn[kmd$cluster!=1,]

# 计算移除年龄离群值前后的标准差以及二者的差值

sd(age\_income\_rm$Age, na.rm = TRUE)

sd(age\_income\_rm2$Age, na.rm = TRUE)

sd(age\_income\_rm$Age, na.rm = TRUE) - sd(age\_income\_rm2$Age, na.rm = TRUE)

# 对移除年龄和收入离群值之后的数据进行聚类分析

set.seed(100)

kmd <- kmeans(age\_income\_rm2n, centers=3)

# 查看聚类后每群的中心

kmd$centers

Income Age

1 0.07142857 0.2975000

2 0.25595238 0.1015625

3 0.31547619 0.5093750

# 作图显示聚类结果

plot(age\_income\_rm2n, col=kmd$cluster)

points(kmd$centers, col=1:3, pch=8, cex=2)

# 显示聚类的分组结果并进行列联表分析, 每群中的数据个数大致相等

kmd$cluster

table(kmd$cluster)

# 对聚类的结果进行特征描述

# Cluster Description

# 查看群中心, 但是得到的是进行过标准化后的群中心, 不是以原始的单位显示的群中心

kmd$centers

Income Age

1 0.07142857 0.2975000

2 0.25595238 0.1015625

3 0.31547619 0.5093750

# 从没有标准化的原始数据中读取分群的结果

clust <- list()

clust[[1]] <- age\_income\_rm2[kmd$cluster==1,]

clust[[2]] <- age\_income\_rm2[kmd$cluster==2,]

clust[[3]] <- age\_income\_rm2[kmd$cluster==3,]

for (i in 1:3){

clust[[i]] <- age\_income\_rm2[kmd$cluster==i,]

}

clust

[[1]]

Income Age

11 30 33

12 60 34

13 35 36

14 55 38

15 40 40

16 50 42

17 45 44

18 45 45

19 50 37

20 40 39

[[2]]

Income Age

1 70 15

2 90 17

3 80 20

5 75 24

6 95 26

7 85 28

8 85 30

9 90 25

[[3]]

Income Age

21 80 47

23 85 51

24 95 53

25 120 55

26 90 57

27 95 59

28 110 61

29 95 63

# 使用原始数据来计算聚类后每一组年龄和收入的平均值

results <- data.frame()

for (i in 1:3){

out <- data.frame(Cluster = i,Income = mean(clust[[i]]$Income), Age = mean(clust[[i]]$Age))

results<- rbind(results, out)

}

# 以分组前的数据计算分组后每一组中年龄和收入的平均值

results

Cluster Income Age

1 1 45.00 38.800

2 2 83.75 23.125

3 3 96.25 55.750

par(mfcol=c(2,6))

# 查看分组后数据分布的情况. 上半部分分别为群体1,2,3的年龄和收入分布, 下半部分是所有群体的年龄和收入分布. 在做图时需要设定年龄和收入的上下限, 最好设置成和母体的上下限一样, 这样才能看出来每一组的收入和年龄在每体中是处于什么水平的.

for (i in 1:3){

hist(clust[[i]]$Age, main = paste("Histogram of Age for Cluster=",i,sep=''), xlab = "Age", xlim = c(15,63), ylim = c(0, 10), nclass=3)

hist(age\_income\_rm2$Age, main = "Histogram of Age for Population", xlab = "Age", xlim = c(15,63), ylim = c(0, 10), nclass=8)

hist(clust[[i]]$Income, main = paste("Histogram of Income for Cluster=",i,sep=''), xlab = "Income", xlim = c(30,120), ylim = c(0, 10), nclass=3)

hist(age\_income\_rm2$Income, main = "Histogram of Age for Population", xlab = "Income", xlim = c(30,120), ylim = c(0, 10), nclass=8)

}

par(mfcol=c(1,1))

# 查看母体年龄和收入的最大最小值

min(age\_income\_rm2$Age)

max(age\_income\_rm2$Age)

min(age\_income\_rm2$Income)

max(age\_income\_rm2$Income)