1 配列 (seriation)

程序包 seriation 中的函数 seriate() 可以对相似度矩阵或距离矩阵的变量重新排序（行和列同时重新排序），使得重排后相似度高的变量彼此靠近。但 seriate() 函数包含若干不同的配列方法，难以选择（效果似乎也不太好）。我们下面仅使用 seriation 包中的绘图函数 pimage, 配列方法使用第 18 讲的单向谱配列（P25）和双向谱配列（P29）

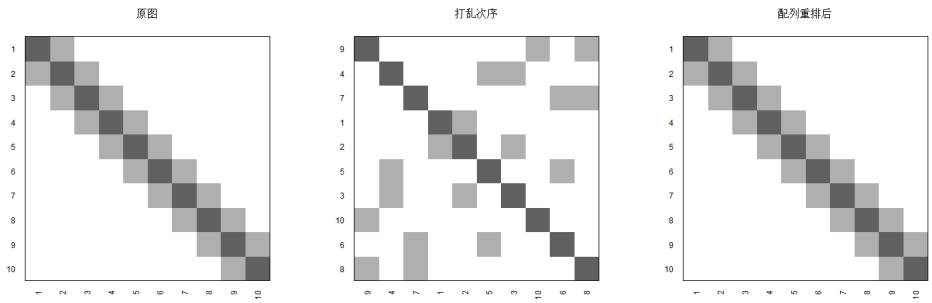
1.1 单向谱配列（函数 seriate1)

```
seriate1=function(S){ #S: similarity matrix
       S=as.matrix(S)
       n=nrow(S)
       apply(S ,1,sum)->d #degree
       diag(d)-S ->L #Laplacian
       eigen(L)[[2]][,n-1]->v #L的最小非0特征根对应的特征向量
       o=order(v) #v的分量的次序（order）
       S.ser =S[o,o] #根据次序重排S.per的行和列
       pimage(S.ser ,col=gray(32:12 /32))
       return(S.ser )
}
```

例 1. 我们生成一个只有主对角和次对角非 0 的相似度矩阵，打乱次序后用 seriate 函数重排次序。

```
library(seriate)
n=10 #set.seed(1)
S=diag(n)
diag(S[-1,][,-n])=1
S=(S+t(S))/2; # S 如右
pimage(S,col=gray(32:0 /32),main="Original similarity")
#S: Original similarity matrix 下图左
per=sample(1:n) ; S.per=S[per,per] #随机置换打乱次序， 下图中
pimage(S.per,col=gray(32:0 /32),main="Permuted similarity")
seriate1(W.per) # seriation重排，下图右
```

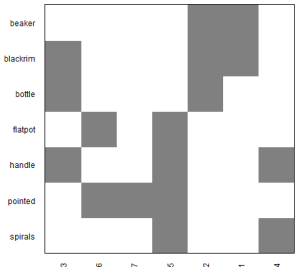
```
 1 1.0 0.5 . . . . . . . .
 2 0.5 1.0 0.5 . . . . . . .
 3 . 0.5 1.0 0.5 . . . . . .
 4 . . 0.5 1.0 0.5 . . . . .
 5 . . . 0.5 1.0 0.5 . . . .
 6 . . . . 0.5 1.0 0.5 . . .
 7 . . . . . 0.5 1.0 0.5 . .
 8 . . . . . . 0.5 1.0 0.5 .
 9 . . . . . . . 0.5 1.0 0.5
10 . . . . . . . . 0.5 1.0
```

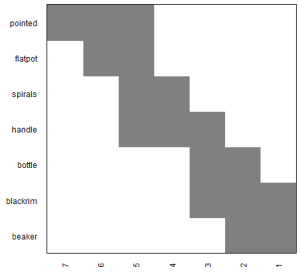


1.2 双向谱配列 (函数 seriate2)

例 2. 第 18 讲 P21 例图，7 个考古地址出土陶器的风格（beaker, balackrim,..., spirals) 矩阵 W 如下表, 其热图如下图左

	3	6	7	5	2	1	4
beaker	0	0	0	0	1	1	0
blackrim	1	0	0	0	1	1	0
bottle	1	0	0	0	1	0	0
flatpot	0	1	0	1	0	0	0
handle	1	0	0	1	0	0	1
pointed	0	1	1	1	0	0	0
spirals	0	0	0	1	0	0	1






```
#输入数据W:
W=matrix(c(0,0,0,0,1,1,0,1,0,0,0,1,1,0,1,0,0,0,1,0,0,0,1,0,1,0,0,0,
1,0,0,1,0,0,1,0,1,1,1,0,0,0,0,0,0,1,0,0,1), 7,7,byrow=T)
rownames(W)=c("beaker","blackrim","bottle","flatpot",
"handle","pointed","spirals")
colnames(W)=c(3,6,7,5,2,1,4)
pimage(W, col=gray(32:16 /32) ) #原始数据W，上图左
```

```
#双向谱配列函数seriate2:
seriate2=function(W){  #W: similarity matrix between two sets of variables
       #W:可以是列联表，丰度矩阵或任何矩阵
       W=as.matrix(W)
       p=nrow(W); q=ncol(W);  n=p+q
       A=matrix(0,n,n)
       A[1:p, (p+1):n]=W;  A[(p+1):n,1:p]=t(W)
       d=apply(A, 1, sum); D=diag(d);  L=D-A
       eigen(L)[[2]][,n-1]->vector
       u=vector[1:p];  v=vector[(p+1):n]
       order(u)->order1;  order(v)->order2
       W[order1,order2]->W.ser
       pimage(W.ser,col=gray(32:16 /32) ) #
       return(W.ser)
} #end

#重排陶器矩阵W的行和列:
seriate2(W) #配列，上图右
```

练习 1. 欧洲语言相似度的另一版本（语言次序打乱的版本）

```
S=read.table("http://staff.ustc.edu.cn/~ynyang/vector/data/Euro-language1.txt")
S=as.matrix(S)
n=nrow(S)
pimage(S , col=gray(32:0 /32) ,key=F)
seriate1(S)
```

试应用 seriate1 函数重排语言，使得相似度高的语言彼此尽量靠近。

2 多维标度法 MDS：cMDS、isoMDS

经典的 MDS 的 R 函数为 cmdscale, 非度量型 MDS 函数为 isoMDS (library(MASS)).

```
cmdscale(d, k=2) # d: symmetric distance matrix or
# or distance object produced by dist or as.dist
isoMDS(d, k=2) # library(MASS)
```

两者都要求输入距离矩阵，数学上距离矩阵是一个 $n \times n$ 方阵，但 R 中的距离矩阵具有特殊的 dist 格式，比如 4 个物件 Item1-4 的两两距离在 R 中表示如下：

```
      Item1 Item2 Item3
Item2   2.4
Item3   2.2   2.7
Item4   3.4   3.6   2.2
```

产生 dist 格式的函数如下

```
d=dist(x,method=) # x: data.frame,
  #method= "euclidean","maximum",
    "manhattan","canberra","binary","minkowski"
d=as.dist(D) #D: 通常的距离矩阵，转换为dist格式
```

前者 (dist) 计算数据矩阵的行之间的距离（dist 格式），后者 (as.dist) 把通常的距离矩阵（方阵）转化成 R 的 dist 格式。

例 **2.** 欧洲 11 种语言的标度方法.

```
S=read.table("http://staff.ustc.edu.cn/~ynyang/vector
/data/Euro-language1.txt",head=T)
lang=rownames(S)
D=(10-S)^0.5 #相似度转化为距离
d=as.dist(D) #R 中距离矩阵是下三角阵的形式
#Classical MDS
mymds= cmdscale(d , k=2) #d: 距离格式，k: 维数
mymds= cmdscale(D, k=2) #D: 一般矩阵，k: 维数
plot(mymds,type="n",xlab="x",ylab="y")
text(mymds,lang,cex=0.8)
#non-metric MDS:
library(MASS)
mynmmds= isoMDS(d,k=2)
plot(mynmmds$points,type="n",xlab="x",ylab="y")
text(mynmmds$points,lang,cex=0.8)
```

练习 2. **第** 13 **讲** P13 **给出了** 6 **门课程** ("Classics","French","English","Math","Pitch","Music") **成绩的相关系数矩阵，我们视之为课程之间的相似系数，**

```
course=c("Classics","French","English", "Math","Pitch","Music")
r=matrix(0,6,6)
lower=c(0.83,0.78,0.70,0.66,0.63,
0.67,0.67,0.65,0.57,
0.64,0.54,0.51, 0.45,0.51, 0.40)
r[lower.tri(r)]=lower
r=r+t(r)
diag(r)=1
rownames(r)=colnames(r)=course
D=sqrt(2*(1-r)) #转化为距离
```

试应用度量型和非度量型的 MDS 方法画出课程的二维坐标表示。

3 聚类 (clustering)

3.1 聚合（agglomeration）层次聚类分析

聚合层次聚类分析的 R 函数为

```
myclust = hclust(d, method="complete")
plot(myclust)
cutree(myclust, k=3) # k: number of clusters
```

其中 d 为距离（相异度）矩阵 (由 dist(x) 计算得到, x 为数据矩阵), method 为连结（linkage）的方法，可选项为 “complete”, “single”, “average”, “centroid”, “median”, “ward.D”, “ward.D2” 等等。使用函数 cutree 可得到聚集成指定类别个数为 k 时的具体的聚类结果，即每个样本点所属的类的标号.

例 **3.** 欧洲城市温度数据. 只使用该数据的前 23 行即欧洲 23 个首都，前 12 列即 12 个月的平均温度（该数据集记为 cap23），试基于 cap23 数据进行完全连结方式的聚合聚类分析:

```
temperature=read.csv("http://staff.ustc.edu.cn/~ynyang/vector/data/temperature.csv",
row.names=1,sep=",")
cap23=temperature[1:23,1:12] #前23行、前12列为23个首都的月温度数据
cap=rownames(cap23)
cap=substr(cap,1,2) #城市名字太长，截取前两个字母
rownames(cap23)=cap
area.true=temperature[1:23,"Area"] # 首都所在的区域
area.true=substr(area.true,1,1)
d=dist(cap23)
mycluster =hclust(d, method="complete")
plot(mycluster )
clusters = cutree(mycluster,k=4) #聚集为k=4类，每个城市所属的类别的标号
> clusters
Am At Be Br Bu Co Du He Ki Kr Li Lo Ma Mi Mo Os Pa Pr Re Ro Sa So St
1 2 3 1 3 3 1 4 3 3 2 1 2 4 4 4 1 3 4 2 3 3 4
#Am,Br, Du,..属于第一类,等等。得到的类别是否与首都所在的区域(area.true)一致?
#注意，考察聚类结果与真实结果是否一致颇具挑战性，比如我们不知道聚类分析得到的
第一类(Am,Br,Du)对应于哪个真实类
tab=table(area.true, clusters)
```

练习 3. 应用基于 averge 连结的聚合层次聚类分析欧洲温度数据。

3.2 K-均值聚类

K-means 聚类法是一种广为流行的聚类方法。假设 $\mathbf{x}_1,...,\mathbf{x}_n \in R^p$, 我们要把这 n 个数据点聚为 K 各类。k-means 方法求解 K 个类的中心 $\mathbf{m}_1,...,\mathbf{m}_K \in R^p$ (以 $\mathbf{m}_k$ 为中心的类记为 C_k), 并把每个数据点分配到距离其最近的中心所对应的类中，目标是使得类内平方和达到最小：

$$SS_W = \sum_{k=1}^{K}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}_k\|^2 1_{(\|\mathbf{x}_i-\mathbf{m}_k\| \le \|\mathbf{x}_i-\mathbf{m}_j\|, j=1,...,k)}$$

该优化问题的经典解法是如下递归迭代算法（Lloyd，1957）：

---

1. 初始化各类中心: 比如随机取数据矩阵的 K 行作为 $\mathbf{m}_k,...,\mathbf{m}_K$ 的初值；
2. 对 $i = 1,2,...,n$, 若 $\|\mathbf{x}_i - \mathbf{m}_k\| \le \|\mathbf{x}_i - \mathbf{m}_j\|, j \ne k, j = 1,...,K$, 则判定 $\mathbf{x}_i \in C_k$ (即将 $\mathbf{x}_i$ 划分到第 k 类)；
3. 求每一类内的平均值 $\bar{\mathbf{x}}_k = \sum_{i \in C_k} \mathbf{x}_i / |C_k|, k = 1,...,K$, 并更新各类的中心：
$$\mathbf{m}_k = \bar{\mathbf{x}}_k, k = 1,...,K.$$
4. 回到 2, 迭代，直到 $\mathbf{m}_k, k = 1,...,K$ 稳定为止。

K-均值聚类的 R 函数为 kmeans:

```
myclust = kmeans(x, centers, nstart) #x: data;
```

其中

- x 为数据矩阵；
- centers 可以指定为类的个数 k, 也可以是 k 个类的中心初值。如果指定类的个数 k，则 k 个类的中心随机选取 x 的 k 行作为中心的初值；
- nstart 为 kmeans 运行次数。因为中心初值选定不同可能会导致不同的结果，所以通常反复运行 kmeans 多次 (nstart)，每次的中心初值都是随机选定的，比较这些结果的目标函数（组内平方和），最小的作为最终结果。

例 **3**（续）欧洲城市温度数据的 k-means 聚类：

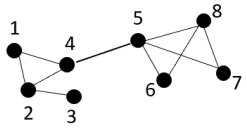
```
temp =read.csv("http://staff.ustc.edu.cn/~ynyang/vector/data/temperature.csv",row.names=1,sep=",")
myclust = kmeans(x=temp[,1:12], centers=4, nstart=50) #4类,重复50次
myclust$centers # 各类的中心
cl = myclust$cluster #各个样本点所属的类别 （1-4）
location=temp[ ,16:15]
plot(location, type="n" )
text(location,rownames(temp ), col=cl ) #四种颜色代表聚成的4类
area.true=temp[,"Area"]
table(area.true,cl) # 错误率是多少?
```

3.3 K-medoid 聚类

K-medoid 聚类分析可以使用 R 软件包 cluster 中的 pam 函数（pam: partition around medoids）。

```
myclust = pam(x, k) #x: data frame or distant matrix;
       #k:number of clusters
```

例 **4**. 下图有边 (edge) 相连的节点认为是相似的



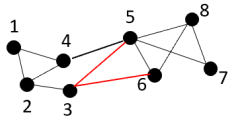
该图的邻接矩阵为

$$A=\begin{pmatrix} 0&1&0&1&0&0&0&0\\ 1&0&1&1&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 1&1&0&0&1&0&0&0\\ 0&0&0&1&0&1&1&1\\ 0&0&0&0&1&0&0&1\\ 0&0&0&0&1&0&0&1\\ 0&0&0&0&1&1&1&0 \end{pmatrix}$$

该矩阵的 (i,j) 元若为 1，表示 i 与 j 是相连的；若为 0，表示它们不连结。这里我们把相连的物体认为是相似的，相似度为 1；不相连的物体的相似度为 0. 我们把 $D=\mathbf{1}\mathbf{1}^{\top}-I_8-A$ 当作相异度/距离矩阵 (把 A 元素 0-1 互换，对角线换成 0)，基于 D 矩阵，应用 k-medoids 方法将 8 个节点聚集为两类:

```
install.packages("cluster")
library(cluster)
a=c(0,1,0,1,0,0,0,0, 1,0,1,1,0,0,0,0, 0,1,0,0,0,0,0,0,
1,1,0,0,1,0,0,0, 0,0,0,1,0,1,1,1, 0,0,0,0,1,0,0,1,
0,0,0,0,1,0,0,1, 0,0,0,0,0,1,1,1,0)
A=matrix(a,8,8)
rownames(A)=colnames(A)=1:8
d=1-A-diag(8)
mypam = pam(d, k=2)
```

练习 4. **例** 4 **中增加两条红色的边**:



试应用 k-medoid **方法将节点聚集为** 2 **类。**

3.4 谱聚类

假设 n 个物体的相似度矩阵为 A，$a_{ij}\geq 0$, 对角元 a_{ii} 不要求一定为 0。定义度数矩阵

$$D=diag(\mathbf{d}),\mathbf{d}=A\mathbb{1},$$

拉普拉斯 (Laplacian) 矩阵定义为

$$L=D-A$$

如果需要将 n 个物体聚集为 K 类，那么我们只需计算 L 的最小 K 个特征根对应的特征向量

$$V = (\mathbf{v}_1, ..., \mathbf{v}_K)$$

并对该矩阵应用 Kmean 是方法（或其他聚类方法）进行聚类。我们可自行定义谱聚类函数 specclust 如下

```
specclust=function(A,k =2) #A:similarity matrix, k: number of clusters
{
      A=as.matrix(A)
      name=rownames(A)
      n=nrow(A)
      d=apply(A,1,sum) #degrees
      D=diag(d)
      L=D-A
      tmp= eigen(L)
      V=tmp$vectors #eigenvectors
      V=V[ ,(n-k+1):n] #V的最后k列

      tmp=kmeans(V, centers=k)
      cluster=tmp$cluster
      names(cluster)=name
      return(cluster)
} #end
```

例如对欧洲语言数据进行谱聚类：

```
S=read.table("http://staff.ustc.edu.cn/~ynyang/vector/data/Euro-language1.txt",head=T)
S=as.matrix(S)

specclust(S, k=4)
 Finnish Danish English German Italian Norwegian French
 4 2 2 2 3 2 3
 Hungarian Polish Dutch Spain
 1 3 2 3
```

练习 5. 将练习 2 的 6 门课程用谱方法聚集成 3 类（注意用相似度而不是距离作为 specclust 函数的输入）。

近些年，基于图谱的聚类方法（称为社区检测）较为流行，下面是简单介绍。

4 附录：图程序包 igraph

igraph 是处理、分析图 (或网络) 数据的 R 包。其中 fastgreedy.community 是节点聚类函数（聚类分析也称作社区检测 community detection）。

```
library(igraph)
par(mfrow=c(2,3),mai=rep(0.3,4))
g1 = graph(edges=c(1,2, 2,3, 3,1, 1,4),directed=T)#边为(1,2),(2,3),(3,1),...
plot(g1,main="g1")

A=matrix(c(0,1,1, 1,0,1, 1,1,0),3,3) #邻接矩阵A
g2=graph_from_adjacency_matrix(A , mode="undirected") #将邻接矩阵A 转化graph格式
plot(g2,main="g2")

g3 = graph_from_literal(a-b-2-c-d-a-e-c, 1-a-2-3-1, d-1---4)
plot(g3,main="g3") #画出网络图

g4 = graph_from_literal(1:2:3-a:b:c)
#二分图(下图1)，1,2,3之间无连结，a,b,c之间无连结
plot(g4, vertex.color=c(1,1,1,2,2,2),main="g4 (bipartite)" )

g5= make_ring(10)
plot(g5, main="g5 (ring)") # 环

# myring2 =add_edges(myring,c(3,8)) #在上图加一条边
# plot(myring2) # 环

g6<- make_tree(40, children = 3, mode = "undirected")
plot(g6, vertex.size=10, vertex.label=NA,main="g6 (tree)") # 树

#聚类/社区检测：
cluster=cluster_leading_eigen(g6)#Newman modularity聚类
cluster = fastgreedy.community(g6) #节点聚类贪心算法
plot(g6,vertex.color=cluster$membership) #颜色标记cluster，如最后一图所示（不同颜色代表不同的社区）
```