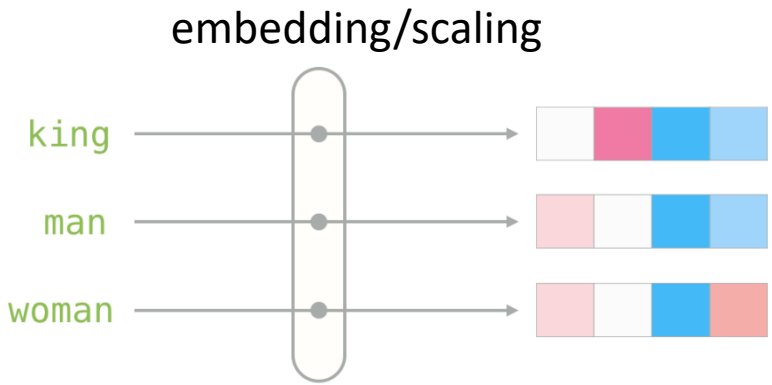
第二十一讲 多维标度法/聚类

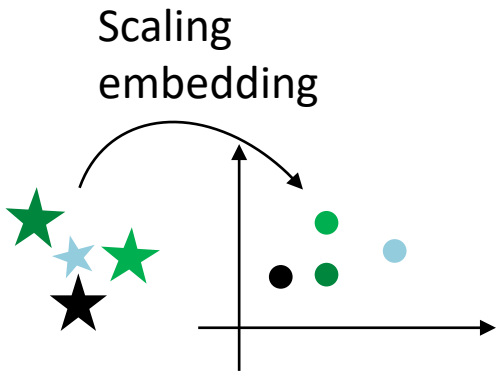
2024.5.27



多维标度法MDS

主观评价或非数字变量的量化方式称为标度方法(scaling)或嵌入(embedding,机器学习)。

多维标度法（MDS，multidimensional scaling）在尽量保持个体/研究对象之间相近程度信息的前提下，用欧氏向量将研究对象表示出来。



单向配列(seriation)问题基于相似度矩阵求解研究对象的一维欧氏表示，并排序。但一维表示信息不够丰富，可能不能正确定代表物体之间的相近关系。

单向配列的多维拓展即是MDS，而单向配列问题可称为1维标度法。

原则上我们可以将第20讲的配列方法向高维拓展(类似于PCA，CCA)，在此不再讨论。下面主要介绍经典/度量MDS方法。

经典或度量型多维标度法（cMDS: Classical MDS ； mMDS: metric MDS）有显式解。cMDS针对相似度，mMDS针对距离。

cMDS也称为主坐标分析方法 (principal coordinates analysis, PCoA) 或 Torgerson Scaling 或 Torgerson-Gower scaling。

cMDS数学框架

给定一个$n\times n$相似度矩阵$S=(s_{ij})$，给定正整数$k\le n$，MDS方法求解$\mathbf{x}_1,...,\mathbf{x}_n\in R^k, X=(\mathbf{x}_1,...,\mathbf{x}_n)^{\mathsf{T}}$，极小化Stress函数：

$$Stress=\sum_{i,j}(s_{ij}-\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j)^2=\|S-XX^{\mathsf{T}}\|_F^2. \qquad (*)$$

cMDS: 1维情形

一维MDS方法求解$x_1,...,x_n\in R^1$，极小化

$$Stress=\sum_{i,j}(s_{ij}-x_ix_j)^2=\sum_{i,j}s_{ij}^{\ 2}-2\sum_{i,j}s_{ij}x_ix_j+\sum_{i,j}(x_ix_j)^2$$

即$\mathbf{x}=(x_1,...,x_n)^{\mathsf{T}}, \lambda=\|\mathbf{x}\|, \mathbf{u}=\mathbf{x}/\lambda,$

$$\text{stress}=\sum_{i,j}s_{ij}^{\ 2}-2\lambda^2\mathbf{u}^{\mathsf{T}}S\mathbf{u}+\lambda^4$$

则使得stress最小的$\mathbf{u}$为S的最大特征根对应的特征向量$\mathbf{u}_1$，最优$\mathbf{x}=\mathbf{u}_1\sqrt{\lambda_1}$.

cMDS: k 维情形

命题1. 假设相似系数矩阵$S \geq 0$，$k \leq r = \text{rank}(S)$, 若$S$的谱分解为

$$S_{n\times n} = U\Lambda U^{\mathsf{T}},$$

其中$U^{\mathsf{T}}U = UU^{\mathsf{T}} = I_n$，$\Lambda = \text{diag}(\lambda_1,...,\lambda_n), \lambda_1 \geq ... \geq \lambda_n \geq 0$为$S$的特征根，则（*）问题中$S$的秩$k$最优逼近为$S_k = U_k\Lambda_k U_k^{\mathsf{T}}$，其中$\Lambda_k = diag(\lambda_1,...,\lambda_k)$，$U_k$为$U$的前$k$列。最优的$n\times k$矩阵$X = U_k\Lambda_k^{1/2}$。

S理解成XX^{T}，X是数据矩阵

如果给定的是欧氏距离矩阵，则我们可以极小化下述stress目标函数，称为度量型MDS：

mMDS

假设$n\times n$距离矩阵$D = (d_{ij})$是欧氏的，给定正整数$k \leq n$, MDS方法求解$\mathbf{x}_1,...,\mathbf{x}_n \in R^k$, $X = (\mathbf{x}_1,...,\mathbf{x}_n)^{\mathsf{T}}$, 极小化Stress函数：

$$Stress = \sum_{i,j}(d_{ij} - \| \mathbf{x}_i - \mathbf{x}_j \|)^2 \qquad (**)$$

解法：根据距离的欧氏表示定理（第19讲命题2），定义相似系数矩阵$S \geq 0$：$S = J\tilde{D}J = (s_{ij})$，$\tilde{D} = (-d_{ij}^2/2)$，其中$J = I_n - \mathbf{1}\mathbf{1}^{\mathsf{T}}/n$。对$S$应用命题1即可。

由命题1，cMDS/mMDS算法如下：

cMDS/mMDS算法

1. 若给定$n\times n$欧氏相似系数矩阵S，goto step 2;
   若给定欧氏距离矩阵$D=(d_{ij})$，则令$S=J\tilde{D}J$，其中$\tilde{D}=\left(-d_{ij}^2/2\right)$,
2. 谱分解$S=U\Lambda U^{\mathsf{T}}$.
3. 对$k<rank(S)$, 取k个最大特征根组成对角阵$\Lambda_k=diag(\lambda_1,...,\lambda_k)$,
   记U的前k列U_k，$X_k=U_k\Lambda_k^{1/2}$，X_k的n个行向量即是R^k中的n个向量。

注1：即使相似系数矩阵或距离矩阵$D=(d_{ij})$不是欧氏的，
我们依然可以通过极小化*Stress*求解,但无显式表达。

注2：非欧氏的距离如何转换为相似系数？
任何距离的减函数都可认为是相似系数，最常用的是Gaussian Kernel：

$$s_{ij}=\exp(-d_{ij}^2/2)$$

R实现

R的经典多维标度法函数为 cmdscale，其输入为距离矩阵。

```
> cmdscale(d, k=2)
  # d: distant matrix, k: low dimension#
```

如果数据是相似系数矩阵，则需要先转化为距离矩阵。

给定相似系数矩阵S,如何转化为距离矩阵D?

根据欧氏距离与欧氏内积的关系（参见19讲命题2的注解），计算

$$\tilde{D} = S - \frac{1}{2}\left(\mathbf{s}\mathbf{1}_n^{\top} + \mathbf{1}_n\mathbf{s}^{\top}\right) = \left(s_{ij} - \frac{s_{ii}+s_{jj}}{2}\right), \quad D = -2\sqrt{\tilde{D}} \quad \text{(元素开根号)}$$

其中$\mathbf{s}$为S的对角元组成的向量。

实践中其它转换方法也可以：

- $D = \max(S) - S, diag(D) = 0$,
- 若$S = R$为相关系数矩阵，则距离矩阵可取为$D = \left(2\sqrt{1-r_{ij}}\right)$。

非度量型MDS (non-metric MDS)

非度量型MDS要求在低维空间上近似地保持原有的相似性或相异性之间的大小次序关系。

对于给定的距离矩阵$D_{n\times n}=(d_{ij})$，假设$d_{i_1j_1}\le....\le d_{i_Nj_N}, N=n(n-1)/2$

非度量型MDS求解$\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n\in R^k$，$\mathbf{x}_i$与$\mathbf{x}_j$的距离记为$\delta_{ij}$，使得$\delta_{i_1j_1}\le...\le\delta_{i_Nj_N}$成立或近似成立(保序：isotonic)。

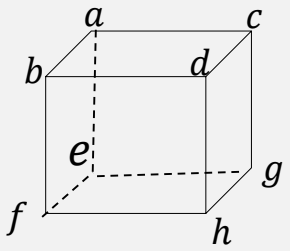
Kruskal - Shepard non - metric scaling:

对于给定的距离矩阵$D=(d_{ij})$，求$\mathbf{x}_1,...,\mathbf{x}_n\in R^k$和单调增函数$f$,极小化

$$Stress(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n;f)=\frac{\sum_{i\neq j}\left(f(d_{ij})-\|\mathbf{x}_i-\mathbf{x}_j\|\right)^2}{\sum_{i\neq j}\|\mathbf{x}_i-\mathbf{x}_j\|^2}.$$

```
library(MASS)
isoMDS(d,  k = 2)
```

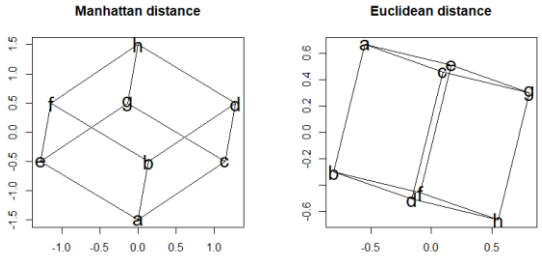
例1. 单位边长三维立方体的8个顶点之间的Manhattan距离矩阵如下：



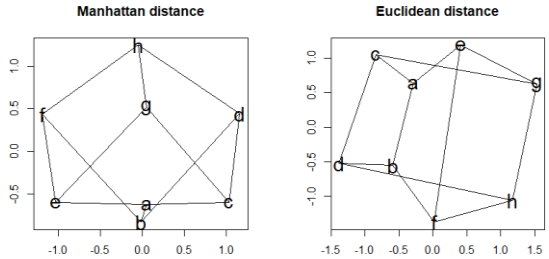
	a	b	c	d	e	f	g	h
a	0	1	1	2	1	2	2	3
b	1	0	2	1	2	1	3	2
c	1	2	0	1	2	3	1	2
d	2	1	1	0	3	2	2	1
e	1	2	2	3	0	1	1	2
f	2	1	3	2	1	0	2	1
g	2	3	1	2	1	2	0	1
h	3	2	2	1	2	1	1	0

应用MDS方法（左列基于Manhattan距离，右列基于欧氏距离矩阵）

cMDS



isoMDS



例2（空间/飞行距离的平面展示，课本Example 12.14），类似地图制作

TABLE 12.7 AIRLINE-DISTANCE DATA

	Atlanta (1)	Boston (2)	Cincinnati (3)	Columbus (4)	Dallas (5)	Indianapolis (6)	Little Rock (7)	Los Angeles (8)	Memphis (9)	St. Louis (10)	Spokane (11)	Tampa (12)
(1)	0											
(2)	1068	0										
(3)	461	867	0									
(4)	549	769	107	0								
(5)	805	1819	943	1050	0							
(6)	508	941	108	172	882	0						
(7)	505	1494	618	725	325	562	0					
(8)	2197	3052	2186	2245	1403	2080	1701	0				
(9)	366	1355	502	586	464	436	137	1831	0			
(10)	558	1178	338	409	645	234	353	1848	294	0		
(11)	2467	2747	2067	2131	1891	1959	1988	1227	2042	1820	0	
(12)	467	1379	928	985	1077	975	912	2480	779	1016	2821	0

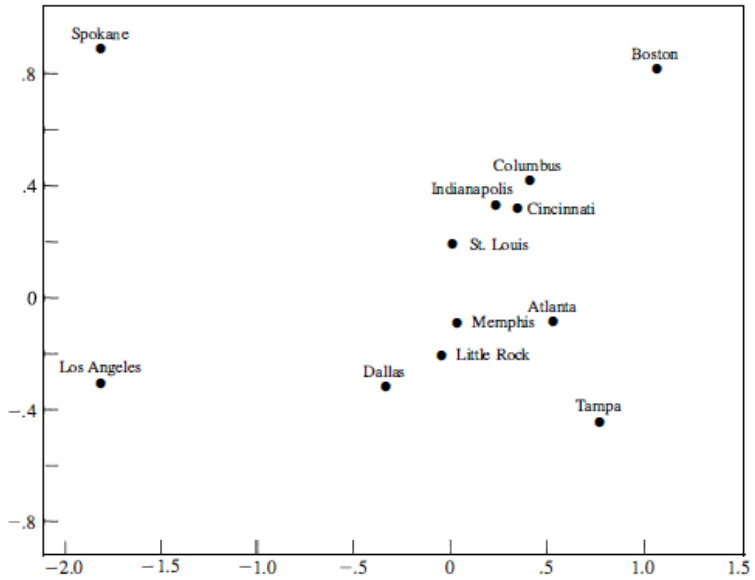


Figure 12.14 A geometrical representation of cities produced by multidimensional scaling.

例3（课本例12.2，11种欧洲语言的比较）各种语言在历史上不断演变或相互影响，研究语言之间的关系有助于了解历史上的文化融合过程。语言的相似性可以体现在多方面，主要体现在发音，其中数字1,2,…,9, 10的读音颇具代表性。下表给出了欧洲11种语言的1-10的写法，它们的相似性如何度量？

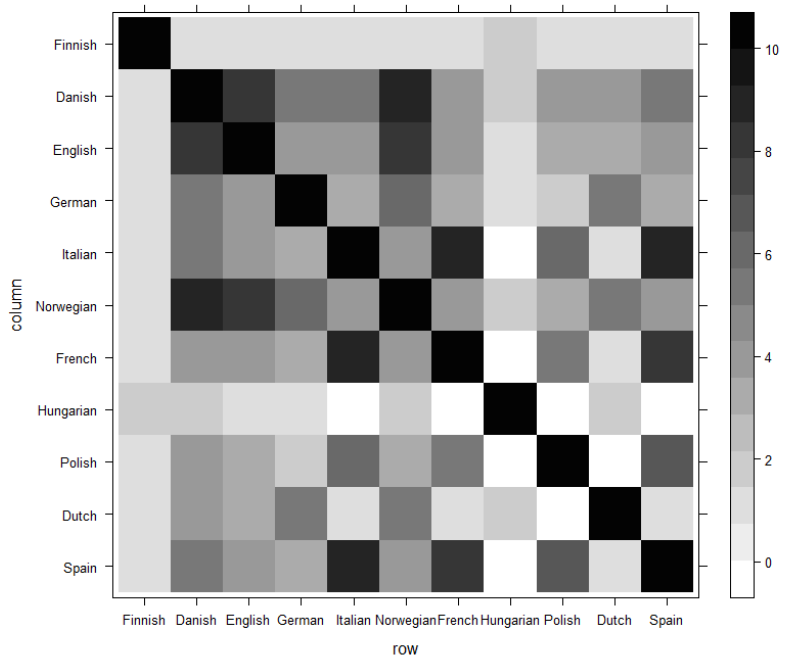
TABLE 12.3 NUMERALS IN 11 LANGUAGES

English (E)	Norwegian (N)	Danish (Da)	Dutch (Du)	German (G)	French (Fr)	Spanish (Sp)	Italian (I)	Polish (P)	Hungarian (H)	Finnish (Fi)
one	en	en	een	eins	un	uno	uno	jeden	egy	yksi
two	to	to	twee	zwei	deux	dos	due	dwa	ketto	kaksi
three	tre	tre	drie	drei	trois	tres	tre	trzy	harom	kolme
four	fire	fire	vier	vier	quatre	cuatro	quattro	cztery	negy	neua
five	fem	fem	vijf	funf	cinq	cinco	cinque	piec	ot	viisi
six	seks	seks	zes	sechs	six	seis	sei	szesc	hat	kuusi
seven	sju	syv	zeven	sieben	sept	siete	sette	siedem	het	seitseman
eight	atte	otte	acht	acht	huit	ocho	otto	osiem	nyolc	kahdeksan
nine	ni	ni	negen	neun	neuf	nueve	nove	dziewiec	kilenc	yhdeksan
ten	ti	ti	tien	zehn	dix	diez	dieci	dziesiec	tiz	kymmenen

我们两两比较11种语言的数字1-10的首字母是否相同，以首字母相同的数字个数作为相似性度量，以首字母不同的个数作为距离。比如英语E 和挪威语N有8个数字的首个字母相同，相似系数为8，距离为2。 下表为11种语言的相似度矩阵：

	German	Italian	Hungarian	Finnish	Spain	Norwegian	Danish	Dutch	French	English	Polish
German	10	3	1	1	3	6	5	5	3	4	2
Italian	3	10	0	1	9	4	5	1	9	4	6
Hungarian	1	0	10	2	0	2	2	2	0	1	0
Finnish	1	1	2	10	1	1	1	1	1	1	1
Spain	3	9	0	1	10	4	5	1	8	4	7
Norwegian	6	4	2	1	4	10	9	5	4	8	3
Danish	5	5	2	1	5	9	10	4	4	8	4
Dutch	5	1	2	1	1	5	4	10	1	3	0
French	3	9	0	1	8	4	4	1	10	4	5
English	4	4	1	1	4	8	8	3	4	10	3
Polish	2	6	0	1	7	3	4	0	5	3	10

下图是相似度矩阵的热图，颜色越深相似度越大。如何在欧氏空间用向量将这些语言表示出来？我们尤其关心一维（排序）和二维表示。



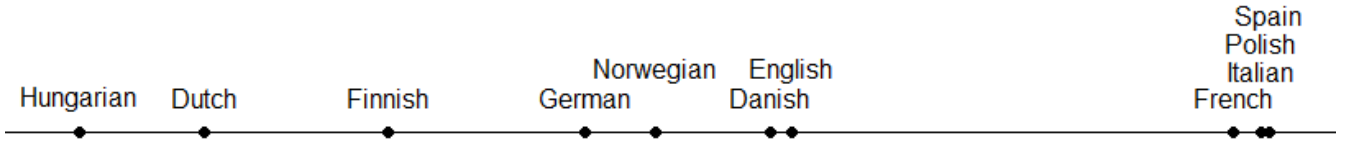
一维MDS/seriation:

求解一维欧氏空间（数轴）的坐标表示，使得这些数轴上的点近似代表各个变量（语言），因此这些数字的大小代表了各语言的次序。

一维 cMDS

11种语言对应的一维cMDS坐标（实数）为

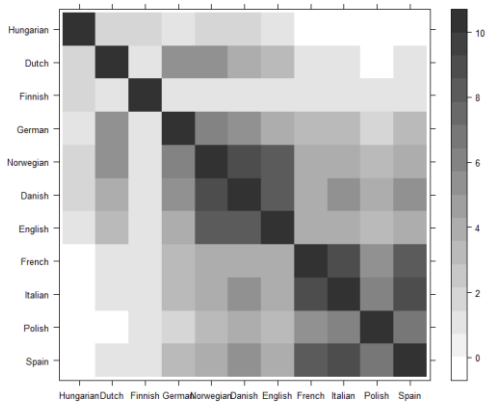
German	Italian	Hungarian	Finnish	Spain	Norwegian	Danish	Dutch	French	English	Polish
-1.42	3.69	-5.24	-2.91	3.76	-0.88	-0.02	-4.3	3.49	0.14	3.69



日耳曼语系

这几种语言高度相似。除了波兰语都属于拉丁语系.

相隔越远的点越不相似，越近的点越相似，因此这些点是有次序的。依据该次序重排相似度矩阵得到下图。

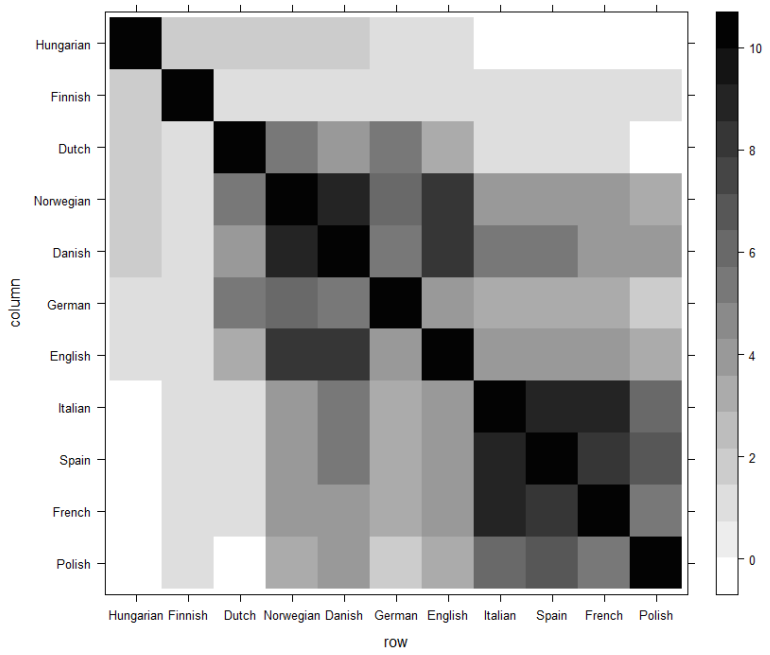


单向谱配列

应用第20讲单向谱配列方法得到拉普拉斯矩阵的最小非0特征根的特征向量：

Polish	Italian	French	Spain	English	German	Danish	Norwegian	Dutch	Finnish	Hungarian
-0.2	-0.18	-0.18	-0.18	-0.12	-0.11	-0.1	-0.1	-0.03	0.39	0.81

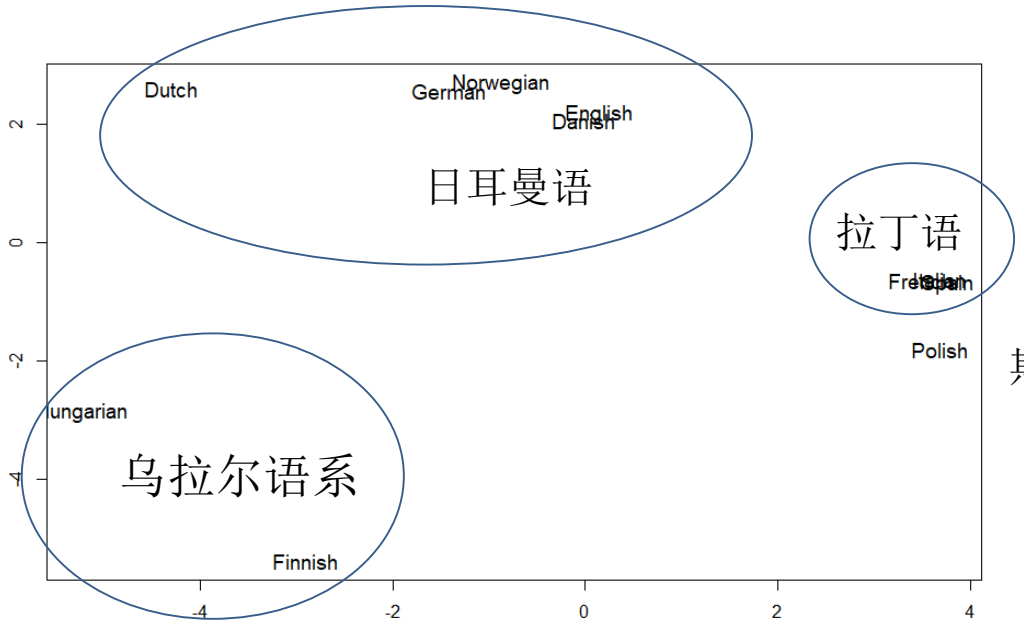
结果与一维cMDS类似，但Hungarian与Finnish更接近，而不是Dutch， 这或许更合理一些。



二维cMDS

二维cMDS能得到更多信息： 二维空间中Polish与拉丁语系不再非常接近，Dutch与Hungarian，Finnish不再接近。

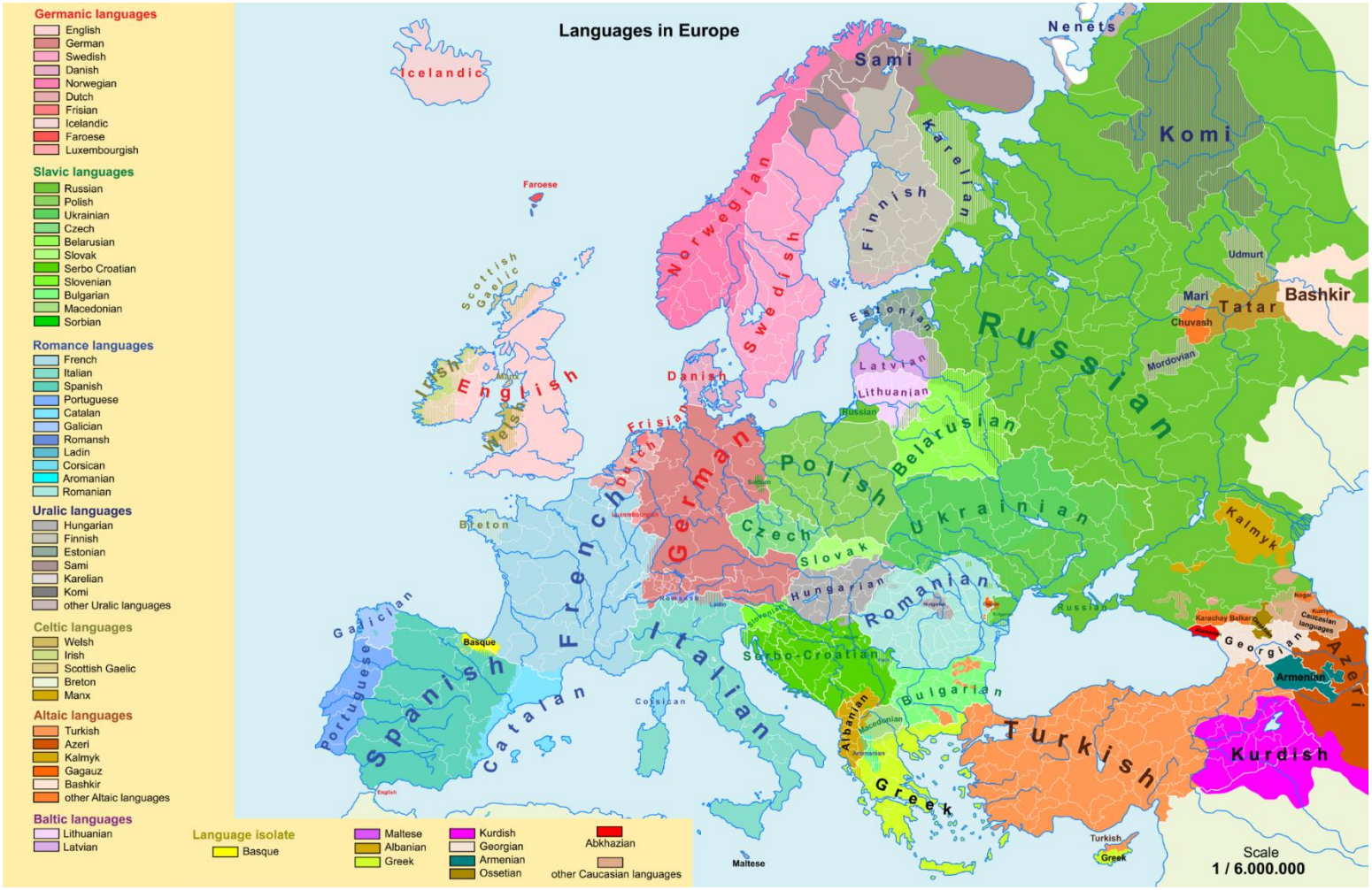
x坐标次序与上页一维坐标次序基本相同。



斯拉夫：Polish

Hungarian，Finnish属于乌拉尔语系, 不属于印欧语系，具有马扎尔人（匈人）的特征，与其它欧洲语言差别较大。

欧洲语言地图


Languages in Europe
Germanic languages
English
German
Swedish
Danish
Norwegian
Dutch
Frisian
Icelandic
Faroese
Luxembourgish
Slavic languages
Russian
Polish
Ukrainian
Czech
Belarusian
Slovak
Serbo Croatian
Slovenian
Bulgarian
Macedonian
Sorbian
Romance languages
French
Italian
Spanish
Portuguese
Catalan
Galician
Romansh
Ladin
Corsican
Aromanian
Romanian
Uralic languages
Hungarian
Finnish
Estonian
Sami
Karelian
Komi
other Uralic languages
Celtic languages
Welsh
Irish
Scottish Gaelic
Breton
Manx
Altaic languages
Turkish
Azeri
Kalmyk
Gagauz
Bashkir
other Altaic languages
Baltic languages
Lithuanian
Latvian
Language isolate
Basque
Maltese
Albanian
Greek
Kurdish
Georgian
Armenian
Ossetian
Abkhazian
other Caucasian languages
Scale
1 / 6.000.000

附录：tSNE：t-distributed stochastic neighbor embedding

MDS以及其它传统方法在低维空间展示高维数据的时候，经常容易出现拥挤现象， Maaten and Hinton (2008)提出了tSNE方法。

tSNE

给定n个物体(*object*)的距离矩阵$D=(d_{ij})$,可以是主观距离，但通常是高维空间中n个点$\mathbf{x}_i, i=1,...,n,$的欧氏距离

$$d_{ij}^2 = \| \mathbf{x}_i - \mathbf{x}_j \|^2 / \sigma_i^2,$$

d_{ij}^2理解为$\mathbf{x}_j$到$\mathbf{x}_i$的距离，注意σ_i^2下标为i,允许$d_{ij}^2 \neq d_{ji}^2$。定义物体j与i的相似性度量(d_{ij}^2的减函数):

$$p_{j|i} = c_i \exp(-d_{ij}^2 / 2)$$

$\exp(-x^2/2)$ ： Gaussian kernel。kernel=similarity

将其归一化：取$c_i = 1/\sum_{j\neq i} \exp(-d_{ij}^2)$, 则$p_{j|i}$, $j=1,2,...,n$是一个概率分布（$j \neq i$）。

定义$p_{ij} = (p_{j|i} + p_{i|j})/2n$作为物体$i, j$的相似性度量（也称为kernel）, $\sum_{i,j} p_{ij} = 1$。

tSNE求解k维欧氏空间R^k中的n个点$\mathbf{y}_1,...,\mathbf{y}_n$，两两之间的欧氏距离$\tilde{d}_{ij}^2 = \| \mathbf{y}_i - \mathbf{y}_j \|^2$，相似度为

$$q_{ij} = \frac{c}{1+\tilde{d}_{ij}^2},\ c = \left(\sum_{i,j} \frac{1}{1+\tilde{d}_{ij}^2} \right)^{-1}$$

$\frac{1}{\pi(1+x^2)}$: Cauchy分布，t_1分布

$\sum_{i,j} q_{ij} = 1$。我们希望这些相似度q_{ij}与原始相似度p_{ij}尽量接近，

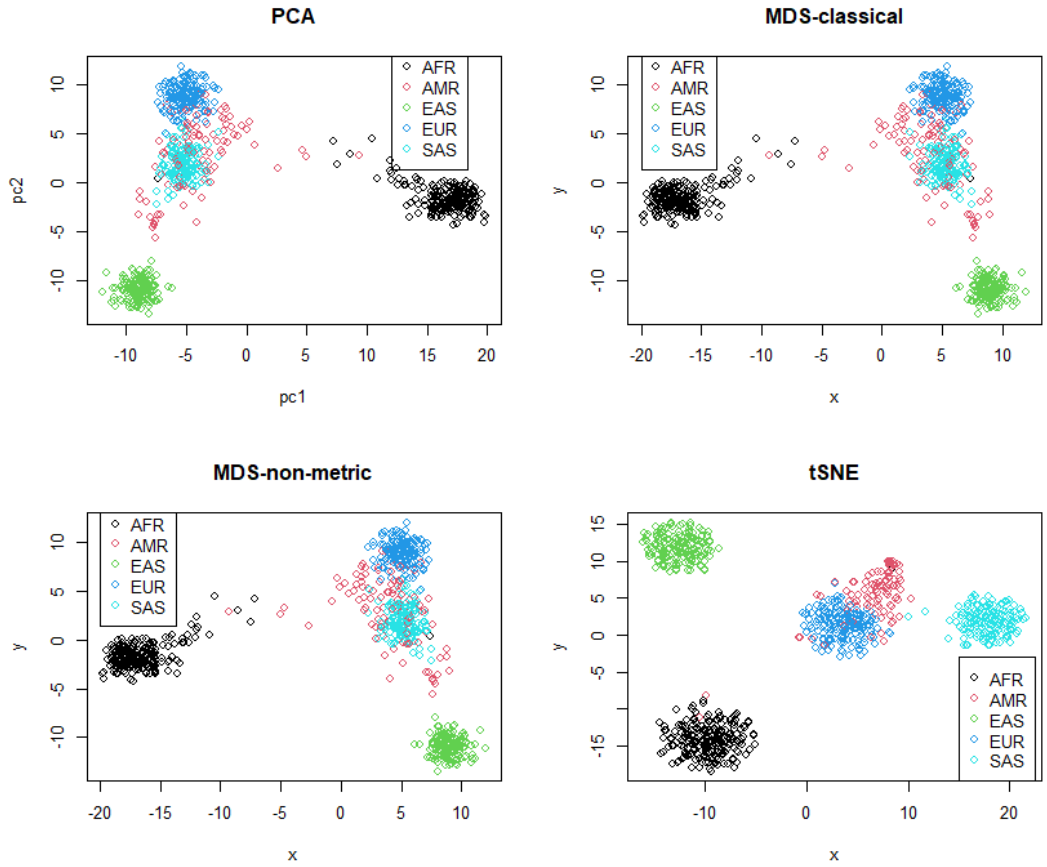
注意两者都是概率分布。KL距离度量两个概率分布的差距：

$$\mathrm{KL}(p \mid q) = \sum_{i,j} p_{ij} \log(p_{ij} / q_{ij})$$

tSNE求解$\mathbf{y}_1,...,\mathbf{y}_n \in R^k$使得$\mathrm{KL}(p \mid q)$极小。

注：SNE(tSNE的前身)中p_{ij}, q_{ij}都取Gaussian kernel，tSNE为避免低维表示中数据点过于拥挤，将q_{ij}修改为Cauchy kernel $q_{ij} \propto \frac{1}{1+\tilde{d}_{ij}^2}$，该分布散度较大。

例6. 基因数据（lab4第1题）的2维展示。PCA和两种MDS效果类似。除了欧洲和美洲之外，其它各种群分离的很好，特别地，tSNE能将SAS与其他种群分开。




```
genedata=read.table("http://staff.ustc.edu.cn/~ynyang/vector/data/genedata1.txt")
genedata[,1]->race
genedata[,-1]->gendata

#PCA
X=scale(genedata,center=T, scale=T)
svd(X)->tmp
u=tmp$u
d=tmp$d
pc1=u[,1]*d[1]
pc2=u[,2]*d[2]
col=as.numeric(as.factor(race))

plot(pc1,pc2, col= col  )
legend(12, 12, legend=c("AFR", "AMR", "EAS", "EUR", "SAS"),col=1:5,pch=1)
title("PCA")

#MDS
dist.gene=dist(genedata)
d=dist.gene
 gene.mds1 =cmdscale(d, k=2) #经典的MDS
plot(gene.mds1,col=col, xlab="x", ylab="y")

library(MASS)
gene.mds2= isoMDS(d,k=2)$points #非度量型MDS
plot(gene.mds2,col=col, xlab="x", ylab="y")

#tSNE:
library(Rtsne)
Rtsne(X )->tmp
plot(tmp$Y, col=race, xlab="x",ylab="y")
```

总结11-20讲

至此（11-20讲），我们介绍了数据的欧氏表示(也称为嵌入)：

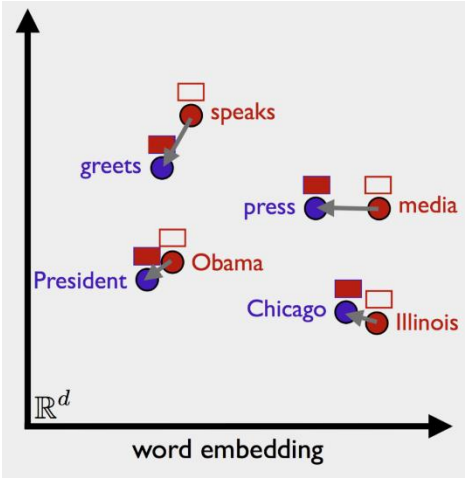
- 高维数据的压缩、低维表示（降维）：PCA，SVD，CCA
- 非欧氏相似度/距离的欧氏表示：MDS

这些方法在机器学习中称为嵌入（embedding），核心是SVD或谱分解，核心矩阵是$X^{\mathsf{T}}X$和XX^{T}：

- PCA基于协方差矩阵或$X^{\mathsf{T}}X$；
- MDS基于相似度矩阵（理解成XX^{T}）；
- SVD基于$X^{\mathsf{T}}X$和XX^{T}。

Embedding 是深度学习的关键技术。在文本处理中(NLP: natural language processing)中

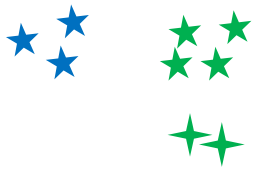
- 首先需要将单词(word)用欧氏向量表示，通常用one-hot方法（哑变量）表示，即只有一个1，且与全是0的向量表示每个单词。
- 结合词汇的相近性和用词环境，进一步利用SVD等技术得到保留相近性的更好的向量表示。最近几年流行的方法是word2vec.



聚类分析Cluster Analysis

聚类分析 Cluster Analysis

聚类分析将相似的或者距离较小的个体聚集成一类(cluster)，不相似的个体分属不同的类。



聚类分析在类别(cluster)未知的数据中挖掘出分类信息的方法， 是一种无监督学习方法 （如果数据已知分类信息，则称为是有监督的）。

目标：n个物件划分为K个类，共有不同的分法：

$$1/K!\sum_{j=0}^{K}(-1)^{K-j}\binom{K}{j}j^n = O(K^n)$$

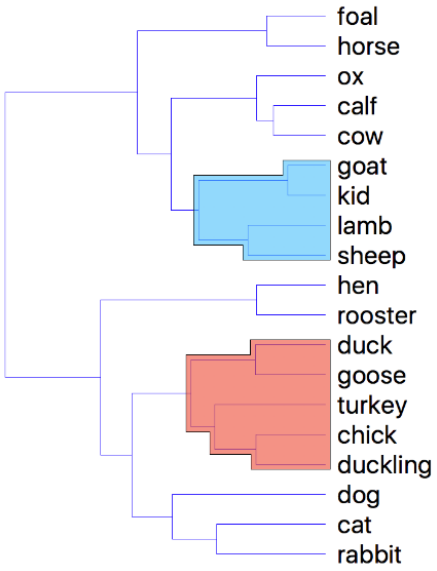
$n=100, K=2, 2^{100}=30$位

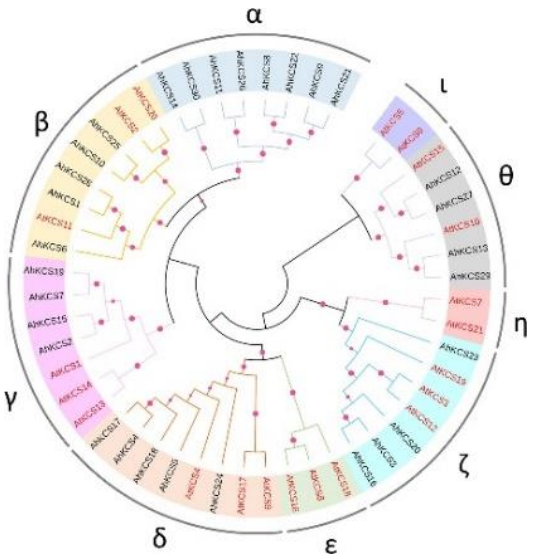
聚类算法基本都是贪心算法。

主要介绍几个经典的算法： 层次聚类、K-medoid聚类、K -均值聚类和混合高斯分布模型 、谱聚类

层次聚类Hierarchical Clustering

层次聚类将研究对象逐步合并（或分拆），也称作系统聚类，主要包括聚合层次聚类法和分割层次聚类法。其中的关键是定义类与类之间的距离。

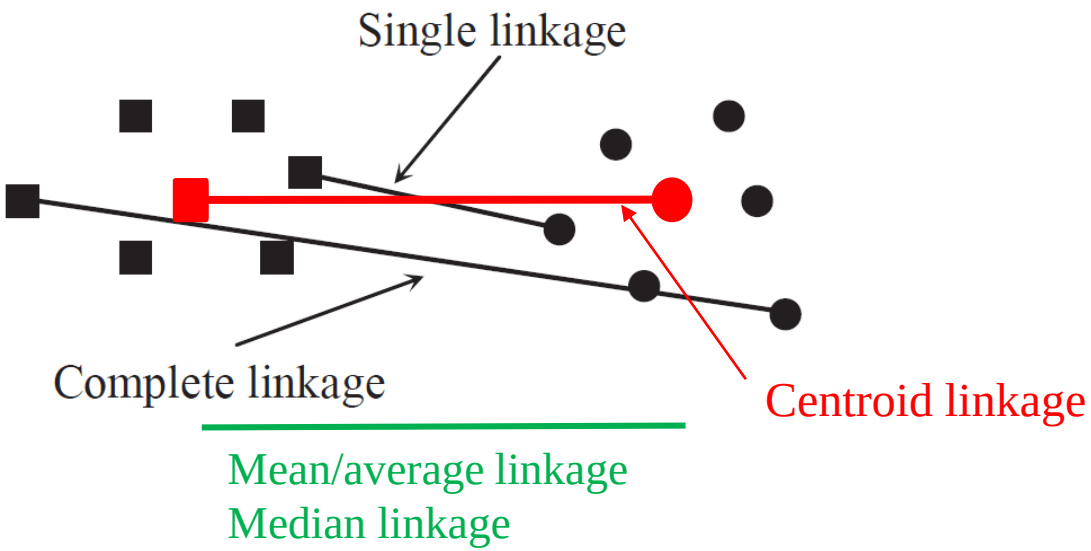




Linkage: 类之间的距离

层次聚类中需要考虑子集(类)的合并或分拆，因此需要定义类之间的距离。距离小的（相似的）子集聚为一类（称为连结 linkage: the manner being united）。 假设个体a, b之间的距离为$d(a, b)$, 则集合之间的距离一般定义为：

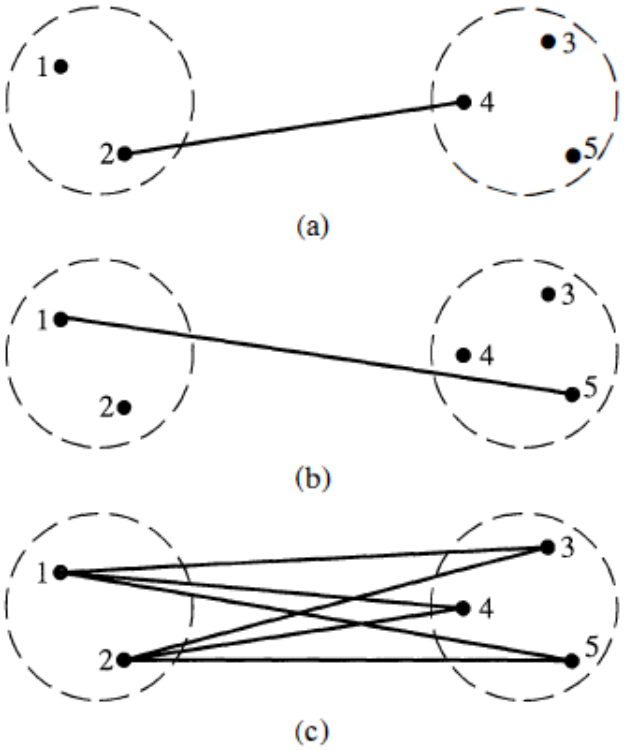
- 单连结(single - linkage)： $d_{\min}(A, B) = \min\{d(a,b), a \in A, b \in B\}$
- 完全连结(complete - linkage)： $d_{\max}(A, B) = \max\{d(a,b), a \in A, b \in B\}$
- 平均连结(average - linkage)： $d_{mean}(A, B) = \sum_{a \in A}\sum_{b \in B} d(a,b)/(|A| \cdot |B|)$
- 其它： centroid, median, Ward linkage



Single: 只要两个集合存在一对点距离较小，就认为这两个集合是同一类。

Complete：只有当两个集合的所有点都距离较小时才认为两个集合属于同一类

Averge：当所有点对之间距离的平均/中位数较小时，认为两个集合属于同一类。median/centroid与此类似。



Cluster distance

d_{24}

d_{15}

$$\frac{d_{13} + d_{14} + d_{15} + d_{23} + d_{24} + d_{25}}{6}$$

聚合层次聚类算法

聚合（agglomeration）层次聚类法首先把每个个体(item,unit)看作一类，然后逐步合并相异度(距离)最小的或相似度最大的类，相继地逐步聚集

1. 初始假定每一个研究对象是一个类(cluster)，计算各个类之间的距离；
2. 距离最小的两个cluster合并成一个更大的cluster；
3. 重新计算各个类(cluster)之间的距离；

重复2，3直到达到某个预定标准或者所有对象合并成一类。

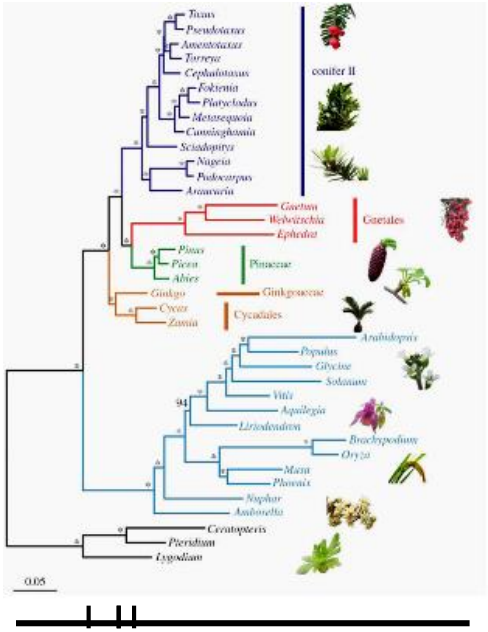
R: hclust(d) # d: 距离矩阵

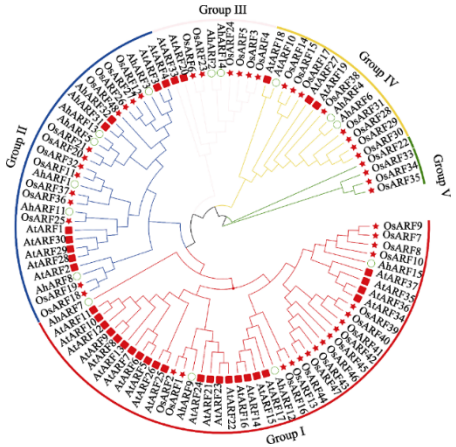
分割层次聚类法

与聚合聚类相反，首先将所有个体当作一类，逐步分割成距离最大或相似度最小的两类 . 效果与聚合方法类似，不常用。

树图

树状图**(Dendrogram)** 表示层次聚类关系，在根部所有物体为一类，朝枝叶方向依次细分（左图）树旁的刻度尺表示聚集或分拆时的距离。更紧凑的树图如右图（不再是树状！）

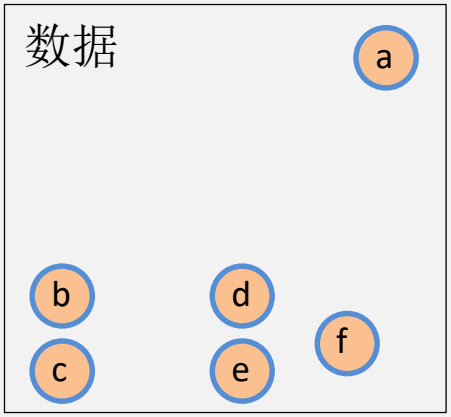





```
myhc= hclust(d,method=) #层次聚类
plot(myhc) #画树图
```

```
library(ape)
plot(as.phylo(myhc), type = "fan")
```

例1. 平面上的点a-f



D=

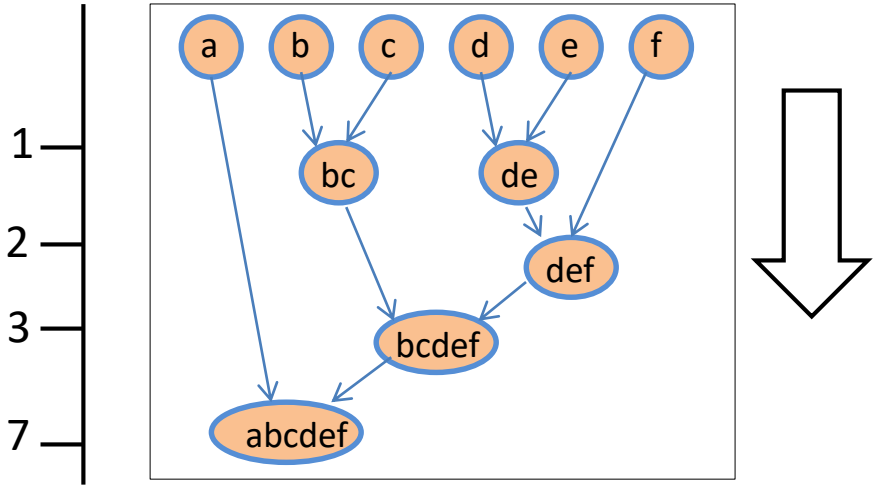
	a	b	c	d	e
b	9				
c	9	1			
d	7	3	4		
e	8	4	3	1	
f	7	5	5	2	2

聚合层次聚类(单连结):

b,c之间、d,e 之间的距离都是1，最小，首先将它们合并集为(bc), (de)两个类。至此有4个类：

(bc), (de), a,f

计算这4个类的两两距离，并合并距离最小的两个类…



交汇处的数字(Height)表示聚合时的距离，比如f与(d,e) 聚合时的距离为2。

例2 (课本例12.3)，5个物体的距离矩阵如右，
Single-linkage 聚集层次聚类分析。

$$\mathbf{D} = \{d_{ik}\} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & & & & \\ 9 & 0 & & & \\ 3 & 7 & 0 & & \\ 6 & 5 & 9 & 0 & \\ 11 & 10 & ② & 8 & 0 \end{bmatrix} \end{array}$$

1. 把每个对象当作一个类，$d_{35} = ② = \min(d_{ij})$, 将3,5合并为一类(35),
   得到4个类：$(35), 1, 2, 4$

计算4个类:(35),1,2,4的两两距离：

$d_{(35)1} = \min(d_{31}, d_{51}) = \min(3,11) = 3,$

$d_{(35)2} = \min(d_{32}, d_{52}) = 7,$

$d_{(35)4} = \min(d_{34}, d_{54}) = 8$

1,2,4之间的距离9,6,5

$$\begin{array}{c} \\ (35) \\ 1 \\ 2 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} (35) & 1 & 2 & 4 \end{array} \\ \begin{bmatrix} 0 & & & \\ ③ & 0 & & \\ 7 & 9 & 0 & \\ 8 & 6 & 5 & 0 \end{bmatrix} \end{array}$$

最小距离 ③ $= d_{(35)1}$

2. 将(35) 和1合并为一类(351).

计算(351),2,4三个类的两两距离(右):

$$d_{(351)2} = \min(d_{(35)2}, d_{12}) = \min(7,9) = 7$$

$$d_{(351)4} = \min(d_{(35)4}, d_{14}) = \min(8,6) = 6$$

$$d_{24} = 5$$

$\Rightarrow$ 最小距离$d_{24} = 5$.

$\Rightarrow$

	(135)	2	4
(135)	0		
2	7	0	
4	6	⑤	0

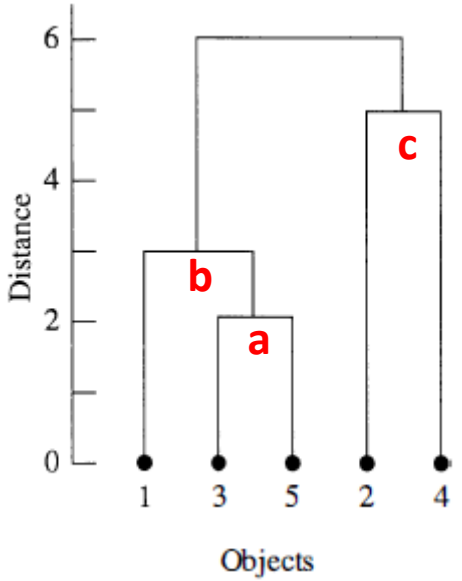
3. 将2 和4合并为一类(24).得到两个类(135),(24)，其距离 = 6.

将上述聚类过程表示为 树图 (dendrogram) 。
从底部开始，按距离从小到大的次序：

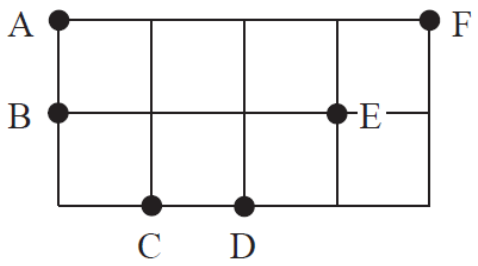
a: 3，5首先合并，距离=2，

b:（35）与1合并，距离=3

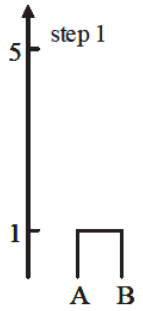
c: 2，4合并，距离=5



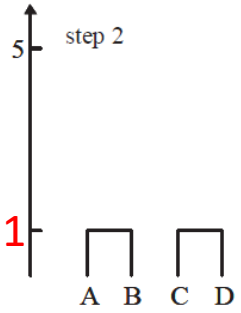
例3 (Complete-linkage聚集层次聚类，Manhattan距离）



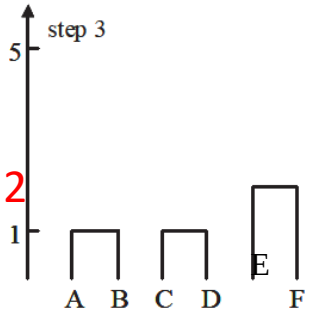
	A	B	C	D	E	F
A	0					
B	1	0				
C	3	2	0			
D	4	3	1	0		
E	4	3	3	2	0	
F	4	5	5	4	2	0



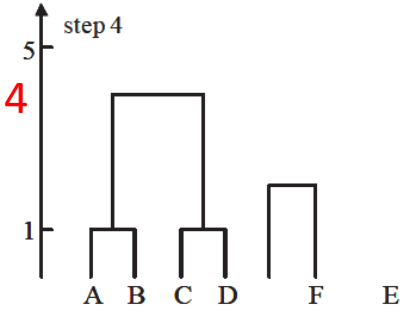
	AB	C	D	E	F
AB	0				
C	3	0			
D	4	1	0		
E	4	3	2	0	
F	5	5	4	2	0



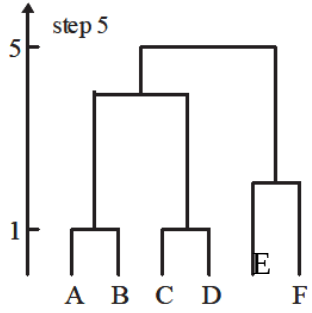
	AB	CD	E	F
AB	0			
CD	4	0		
E	4	3	0	
F	5	5	2	0



	AB	CD	EF
AB	0		
CD	4	0	
EF	5	5	0

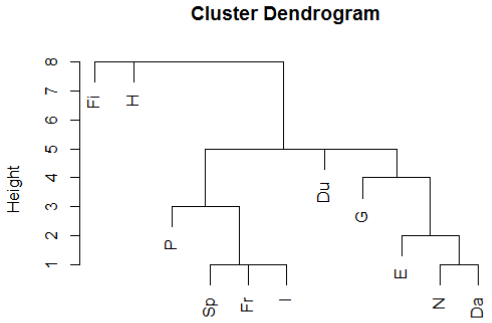


	ABCD	EF
ABCD	0	
EF	5	0

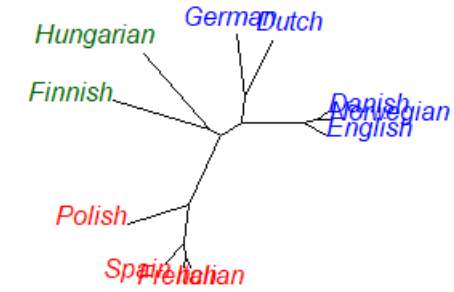


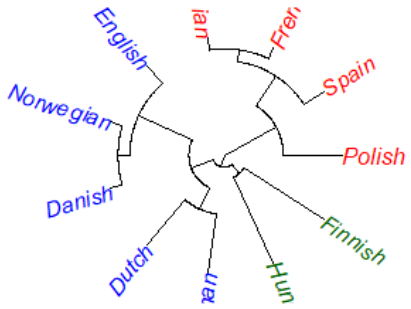
例4（Example 12.3 欧洲语言，R 命令： hclust)

```
> d=10-similarity  # 相似性转为距离
> d=as.dist(d) #
> myclust= hclust(d, method="single")
> myclust= hclust(d, method="complete")
> plot(myclust) #画树图（右）
```




```
#树图的其它画法：
library(ape)
co=c(rep("blue",5), ep("red",4),rep("darkgreen",2))
plot(as.phylo(hc), type = “unrooted”) #右图
plot(as.phylo(hc), type = “fan“,tip.color=co)#右下图
```

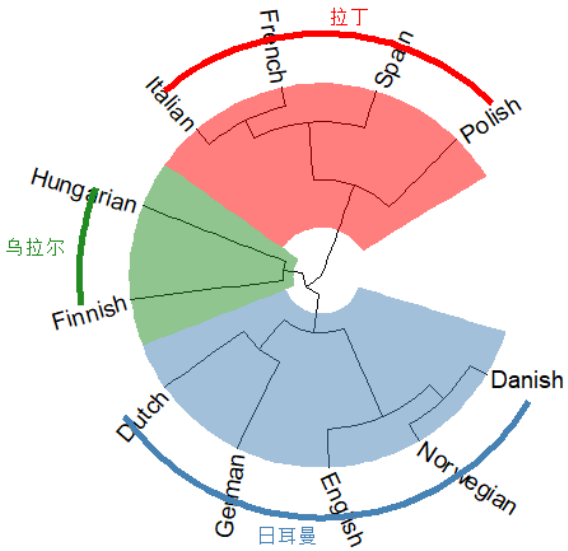




```
install.packages("BiocManager")
BiocManager::install("ggtree")
library(ggtree)
hc=hclust(d)
ggtree(hc,layout="circular")+
 xlim(0,7)+
 geom_tiplab2(offset=0.1,
          size=5)+
 # geom_text(aes(label=node))+

 geom_highlight(node = 13,fill="red")+
 geom_highlight(node=15,fill="steelblue")+
 geom_highlight(node=16,fill="forestgreen")+

 geom_cladelabel(node=13,label="拉丁",
           offset=1.2,barsize = 2,
           vjust=-0.5,color="red")+
 geom_cladelabel(node=15,label="日耳曼",
           offset=1.2,barsize = 2,
           hjust=1.2,vjust=1.5,color="steelblue")+
 geom_cladelabel(node=16,label="乌拉尔",
           offset=1.2,barsize = 2,
           hjust=1.2,color="forestgreen")
```

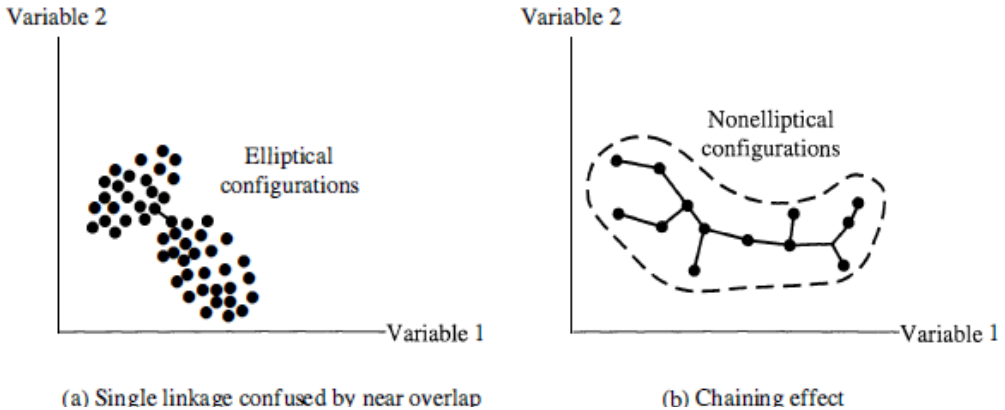


几个评注

单连结链状结构

1. Single-linkage容易形成链状结构, 更适宜于非线性分类：

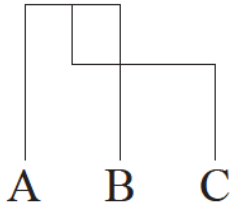
左图两个类分离的较好，但两个类的最近的点相距很近，以此两类的single-linkage距离很小，single-linkage倾向于合并这两类。single-linkage容易形成链状的cluster（右图）。



(a) Single linkage confused by near overlap　　(b) Chaining effect

逆序现象

2. Centroid, median linkage可能会产生逆序(inversion,树倒长)现象。



A,B先合为一类，距离$d(A,B) < d(A,C), d(B,C)$

之后C与(AB)合并，距离$d(C,(AB)) < d(A,B)$

各种连结的优劣比较

- Complete方法以两个类的最远的点之间的距离来决定两个类是否应该聚合或拆分，因而适合类内较为紧密，类之间分离的较好的情形。
- Single方法容易形成长链式的聚集效果，因而对非线性的聚簇数据聚类效果可能较好。
- Average方法介于两者之间。
- Centroid, median 较为稳健，但没有单调性，可能会产生逆序(inversion)现象。
- Ward合并使得组间平方和变化最小的两类。

层次聚类：贪心算法

层次聚类的计算复杂度为$O(n^2 \log(n))$

层次聚类是贪心算法，不一定是最优算法

K-medoid聚类

K-medoid聚类算法只使用距离矩阵进行聚类，各类的中心限定为某个个体，称为medoid。

中心点 Medoid

定义（medoid)：假设n个个体的距离矩阵为$D=(d_{ij})_{n\times n}$，定义medoid为n个点中与其它各点距离之和最小的点：$j_{medoid}=\arg\min_{j\in\{1,2,...,n\}}\sum_{i=1}^{n}d_{ij}$.

划分 Partition

个体指标集$I=\{1,2,....,n\}$的子集$C_1,...,C_K$称为I的一个划分(*partition*)，如果它们两两不交，且$\bigcup_{i=1}^{K}C_i=I$.

K-medoid 聚类问题

假设n个个体的距离为$D=(d_{ij})_{n\times n}$，K-medoid聚类问题求解$\{1,2,....,n\}$的K-划分$C_1,...,C_K$及其中心（*medoids*）$\{m_1,...,m_K\}\subset\{1,2,....,n\}$，使得组内距离之和极小：$\min_{\{C_1,...,C_K\}}\sum_{k=1}^{K}\sum_{i\in C_k}d_{im_k}$

K-medoid 算法

- 当划分已知时，根据中心的定义，容易确定各类的中心；
- 当各类中心已知时，我们只需把每个个体划分到离其最近的中心所在的类；

给定划分或中心的初始值，上述两步递归迭代。

K-medoid clustering算法：

1. 给定$\{1,2,...,n\}$的一个划分$C_1,...,C_K$，对每一个类$1 \le k \le K$，求类内与其它个体总距离最小的点$m_k \in C_k$作为中心：

$$m_k = \arg\min_{i \in C_k} \sum_{j \in C_k} d_{ij}$$

2. 给定当前的中心$\{m_k, k = 1,...,K\}$，将每个个体划分到离它最近的中心所属的类中，即对所有$1 \le i \le n$，求

$$k_i = \arg\min_{1 \le k \le K} d_{im_k}, \quad 1 \le k_i \le K,$$

并将个体i划归到第k_i类。由此得到更新的划分$C_1,...,C_K$.

3. 迭代1-2

R:　pam 函数（package：cluster）

```
> Libr ary(cluster)
> pam(d, k=2)
```

例5（欧洲语言） 欧洲11种语言k-medoid聚类, 应用k-medoids

```
#距离矩阵 d=10-s，s: similarity matrix
> pam(d,4) # k=4  (4类)
Medoids:
    ID
[1,] “2”  “Norwegian” 日耳曼语中心
[2,] “8”  “Italian”  拉丁语中心
[3,] "10" "Hungarian"
[4,] "11" "Finnish"

Clustering vector:
  English  Norwegian    Danish Dutch German
        1          1         1       1       1
French     Spain   Italian    Polish Hungarian   Finnish
     2          2         2         2         3          4
```

Norwegian和Italian分别是日耳曼和拉丁的中心。拉丁源于罗马，日耳曼源于挪威？