Week 6. Hierarchical cluster analysis and K-means algorithm

Hard and soft cluster analysis for unsupervised learning, difference between logistic regression and discriminant analysis. Dissimilarity measures based on Euclidean, Manhattan and Minkowski norms in multidimensional space. Hierarchical clustering (`hclust` function) and dendogram. R libraries `mclust` and `pheatmap`. K-means algorithm for hard clustering and simulation. Statistical derivation of K-means using maximum likelihood function. False clusterization and computing the p-value using simulations. Membership probability and cluster identification using validation observations. Broken-line algorithm for identification of the number of clusters.

R code: `hcl, crime, kmsim`

R functions: `hclust`, `kmeans`

R packagse: `pheatmap, sigclust, dendextend (cutree)`

Literature: `Demidenko_kmeans_2016.pdf`

Data: `T15_1_CITYCRIME.csv`

Introduction

The goal of the cluster analysis (CA) is to identify homogeneous sets of n observations given by the feature vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in R^m$. This is an example of **unsupervised** learning. Remind that discriminant analysis (DA) and logistic regressions belong to supervised learning.

Classification: The difference between DA, LR and CA.

Hard classification means that each object/subject, represented by a feature vector, is assigned to one of the clusters; **soft** classification assigns a *probability* that the objects belongs to a cluster.

Gaussian mixture is an example of **soft** clustering: $\alpha N(\mu_1, \sigma_1^2) + (1-\alpha)N(\mu_2, \sigma_2^2)$.

Hard classification is solved by classical cluster analysis (such as hierarchical or K-means algorithm), **soft** classification is typically solved by Gaussian mixture.

Hierarchical clustering

Agglomerative procedure: add the successive point to the nearest cluster.

R function `hclust`

Dissimilarity measures (distances) between vectors:

1. Euclidean or L_2 norm:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}$$

2. Manhattan or L_1 norm:

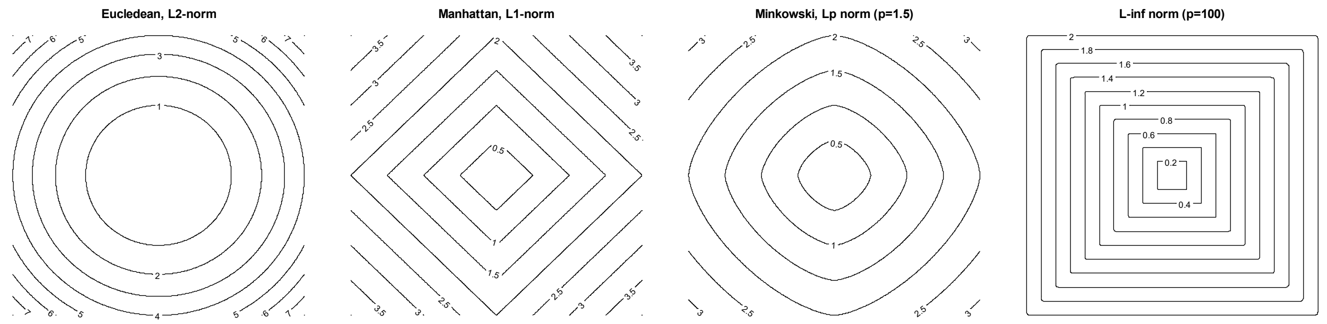
$$d(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j| = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$$

3. Minkowski or L_p norm:

$$d(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|_p = \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p \right)^{1/p}, \quad p > 0$$

4. L_∞ norm

$$d(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|_\infty = \lim_{p \in \infty} \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p \right)^{1/p} = \max_{k=1,..,m} |x_{ik} - x_{jk}|$$



Dissimilarity measures between clusters (cluster C_1 and cluster C_2, d is a dissimilarity/distance between vectors) :

1. Single linkage:

$$\min_{i \in C_1, j \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

2. Complete linkage:

$$\max_{i \in C_1, j \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

3. Average linkage:

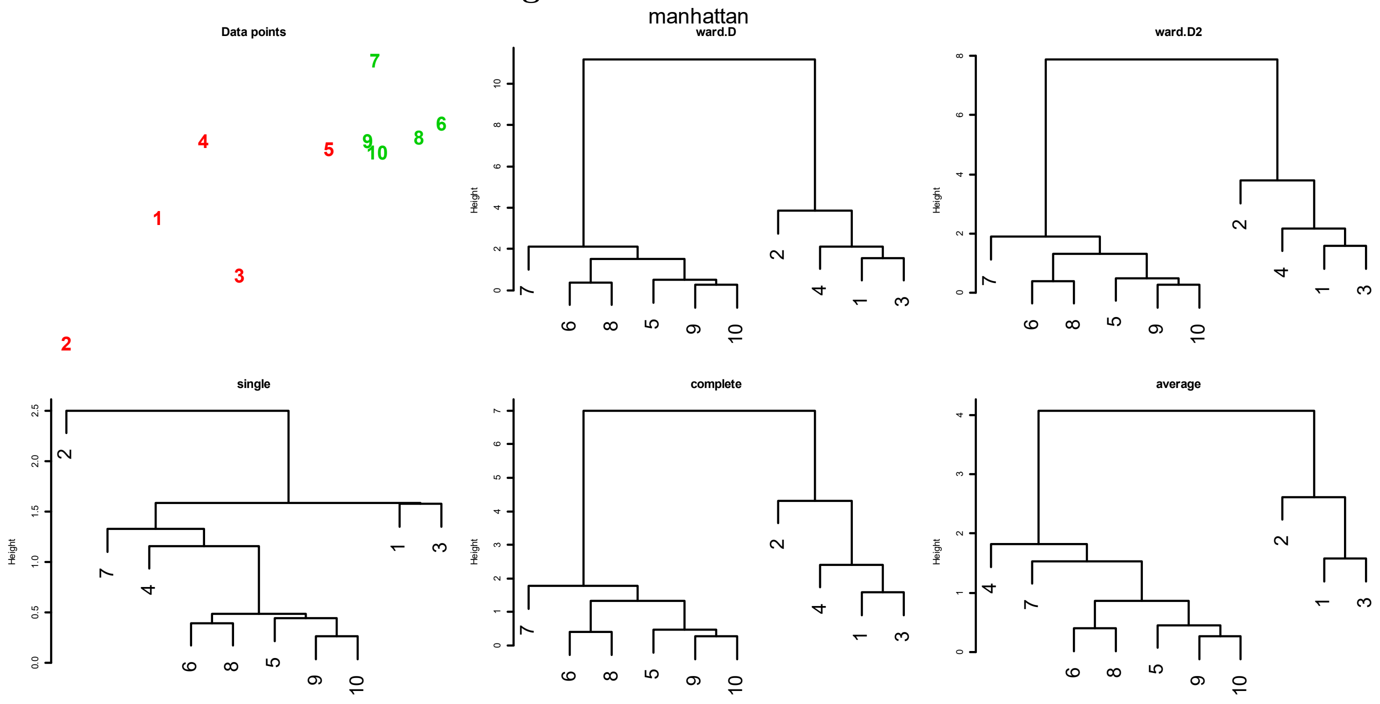
$$\frac{1}{|C_1|\,|C_2|} \sum_{i \in C_1} \sum_{j \in C_2} d(\mathbf{x}_i, \mathbf{x}_j)$$

R function hclust

Below is an educational R code which illustrates how hclust works for different methods of hierarchical clustering.

```
hcl=function(dr="c",n=10,dm=2,distmeth=1,p=1,ss=3)
{
dump("hcl",paste(dr,":\\QBS124\\hcl.r",sep=""))
set.seed(ss)
namd=c("euclidean", "maximum", "manhattan", "canberra", "binary","minkowski")
par(mfrow=c(2,3),mar=c(1,4,3,1),omi=c(0,0,.1,0))
X1=matrix(rnorm(2*n),ncol=2,nrow=n)
X2=matrix(rnorm(2*n,mean=dm),ncol=2,nrow=n)
X=rbind(X1,X2)
d=dist(X,method=namd[distmeth])
plot(X[,1],X[,2],main="Data points",type="n",axes=F,xlab="",ylab="")
text(X[,1],X[,2],1:(2*n),font=2,cex=2,col=c(rep(2,n),rep(3,n)))
meth=c("ward.D", "ward.D2", "single", "complete", "average")
for(i in 1:5)
{
 o=hclust(d,method=meth[i])
 plot(o,main=meth[i],sub="",cex=2,lwd=3)
}
mtext(side=3,namd[distmeth],cex=1.5,outer=T,line=-1)
}
```

Dendogram for manhatten distance



1 City crime clustering

See R function `crime`.

1.1 R package 'pheatmap'

Clustering of observations and variables/features: transpose the data matrix if you wnat to cluster features.
Run `hcl(job=2)`

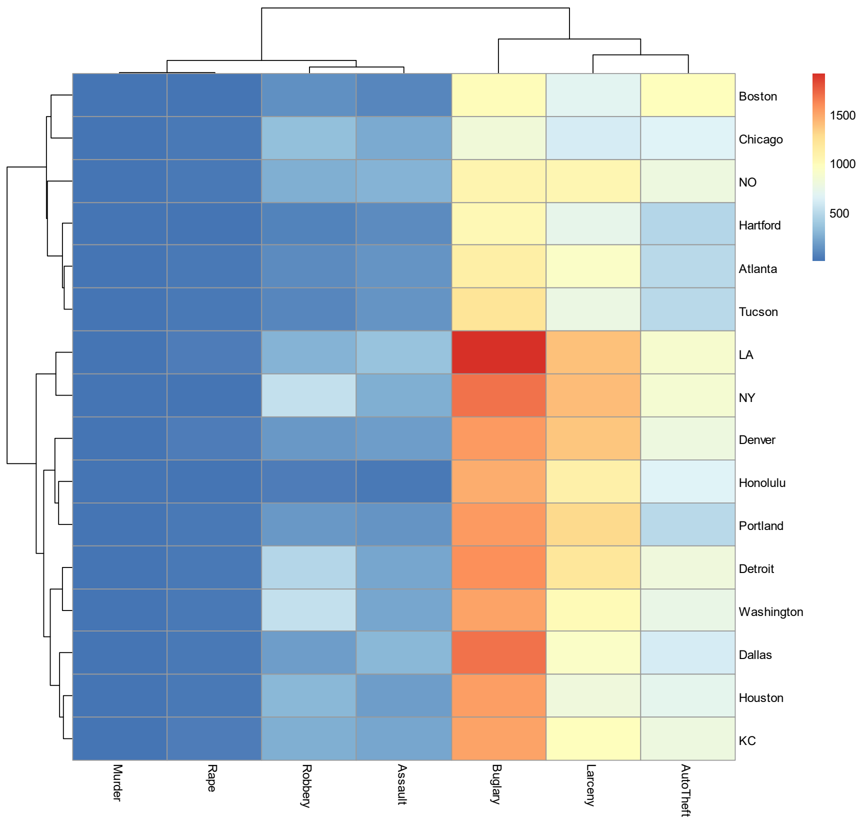
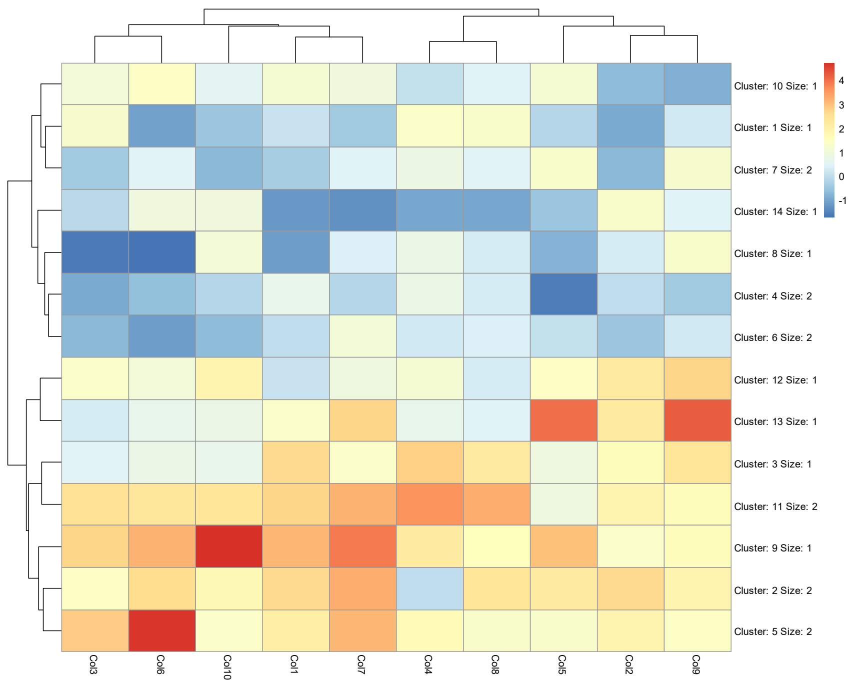
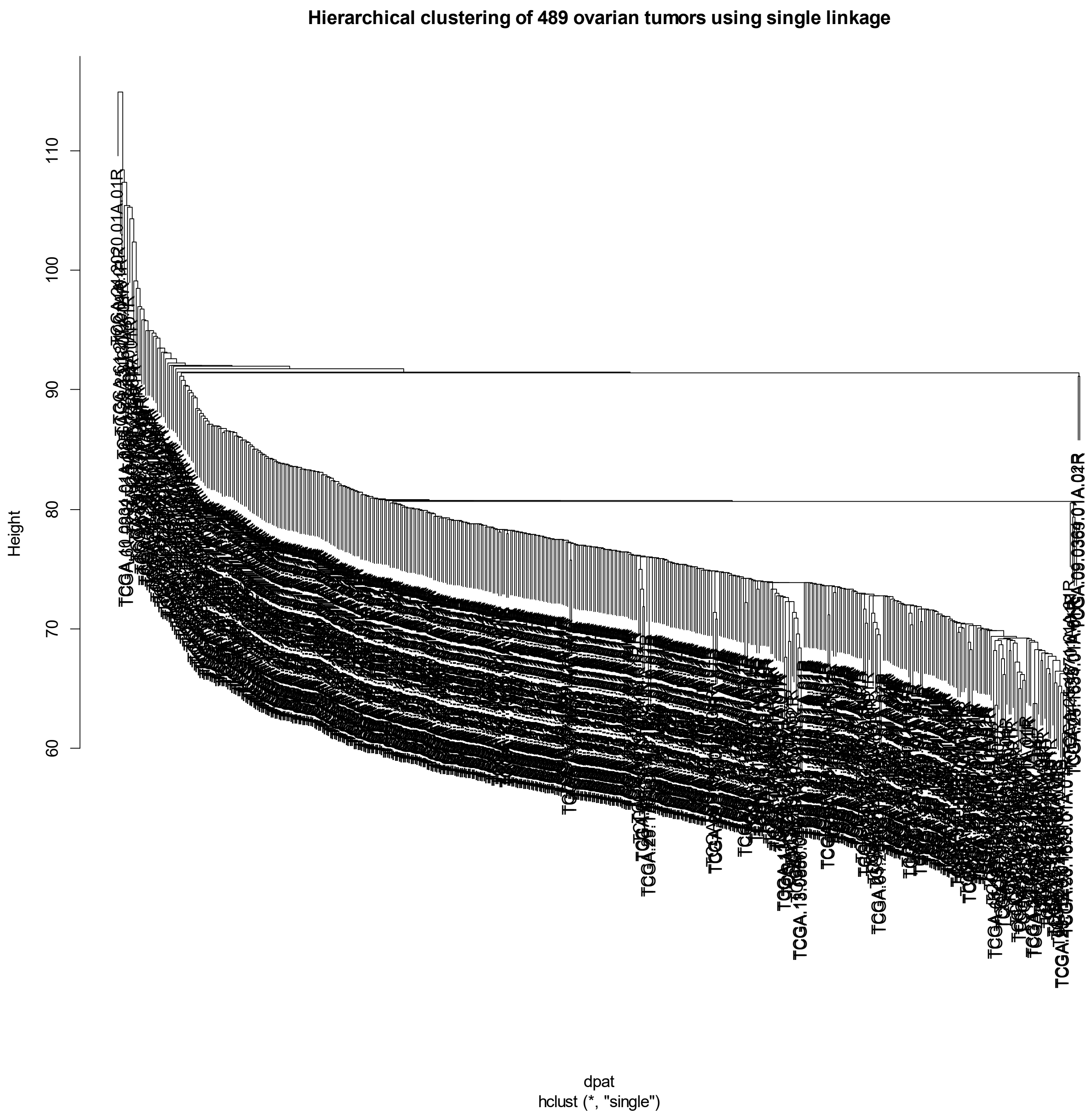


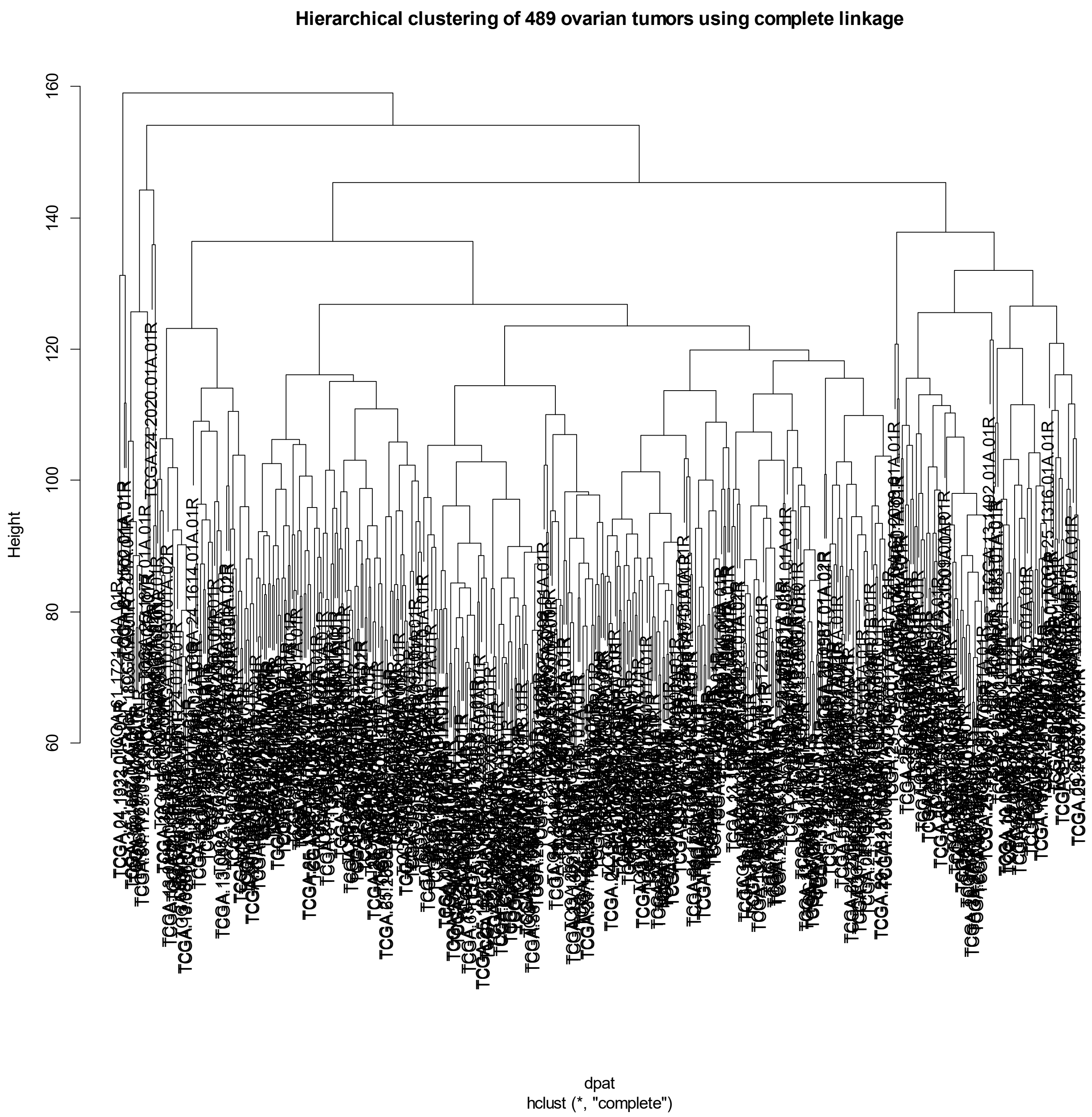
Figure 1:



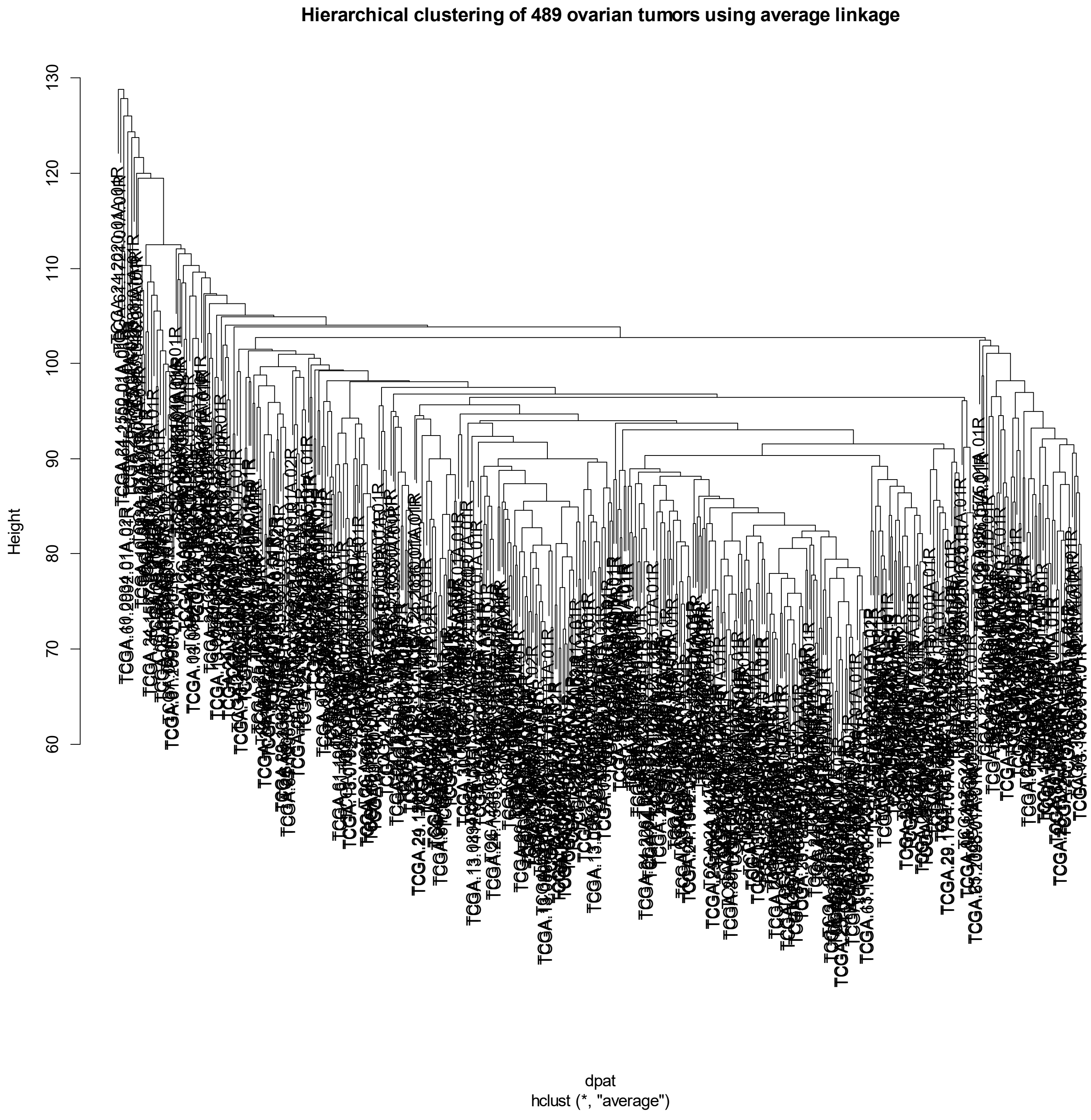
Run `crime(job=2)`

Example 1 *TCGA tumor classification using hierarchical clustering*





Hierarchical clustering of 489 ovarian tumors using complete linkage
Height
160
140
120
100
80
60
dpat
hclust (*, "complete")


Hierarchical clustering of 489 ovarian tumors using average linkage
Height
60
70
80
90
100
110
120
130
dpat
hclust (*, "average")

K-means algorithm

`Demidenko_kmeans_2016.pdf`

It is assumed that n independently distributed observation vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in R^m$ are independent and belong to K groups specified by the index sets $C_1, C_2, ..., C_K$. The number of clusters, K is **known**. These index sets partition the set $\{1, 2, ..., n\}$, so that $\cup_{k=1}^K C_k = \{1, 2, ..., n\}$ and $C_k \cap C_l = \varnothing$ for $k \neq l$.

R function `kmeans`. The optimization is hard because the criterion function it is not smooth, optimization on finite sets. Several starts are required to be sure that it converges to the global minimum.

The criterion of clusterization is

$$\min_{\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_K; C_1,...,C_K} \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 ,$$

typically a Euclidean distance. Since

$$\widehat{\boldsymbol{\mu}}_k = \overline{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{x}_i$$

where n_k is the number of vectors in set C_k, we get rid of the $\boldsymbol{\mu}$s (centers),

$$S_K = \min_{C_1,...,C_K} \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2 .$$

It is called the total within sum of squares. The following sum of squares decomposition holds for any $\{C_1, C_2, ..., C_K\}$:

$$\text{Total SS} = \text{Within SS} + \text{Between SS}$$

or

$$\sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}\|^2 = \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2 + \sum_{k=1}^K n_k \|\overline{\mathbf{x}}_k - \overline{\mathbf{x}}\|^2 ,$$

where $\overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, the center of all data.

Proof. We have

$$\begin{aligned} \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}\|^2 &= \sum_{k=1}^K \sum_{i \in C_k} \|(\mathbf{x}_i - \overline{\mathbf{x}}_k) - (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k)\|^2 = \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2 + \sum_{k=1}^K n_k \|\overline{\mathbf{x}}_k - \overline{\mathbf{x}}\|^2 \\ &\quad -2 \sum_{k=1}^K \sum_{i \in C_k} (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k)'(\mathbf{x}_i - \mathbf{x}_k). \end{aligned}$$

Prove that the third term vanished:

$$\sum_{k=1}^K \sum_{i \in C_k} (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k)'(\mathbf{x}_i - \overline{\mathbf{x}}_k) = \sum_{k=1}^K (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k)' \sum_{i \in C_k} (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k).$$

But

$$\sum_{i \in C_k} (\overline{\mathbf{x}} - \overline{\mathbf{x}}_k) = n_k \overline{\mathbf{x}} - n_k \overline{\mathbf{x}}_k = 0.$$

Therefore the third term is zero.

```
kmeans(x, centers, iter.max = 10, nstart = 1, algorithm =
        c("Hartigan-Wong","Lloyd","Forgy","MacQueen"), trace=FALSE)
```

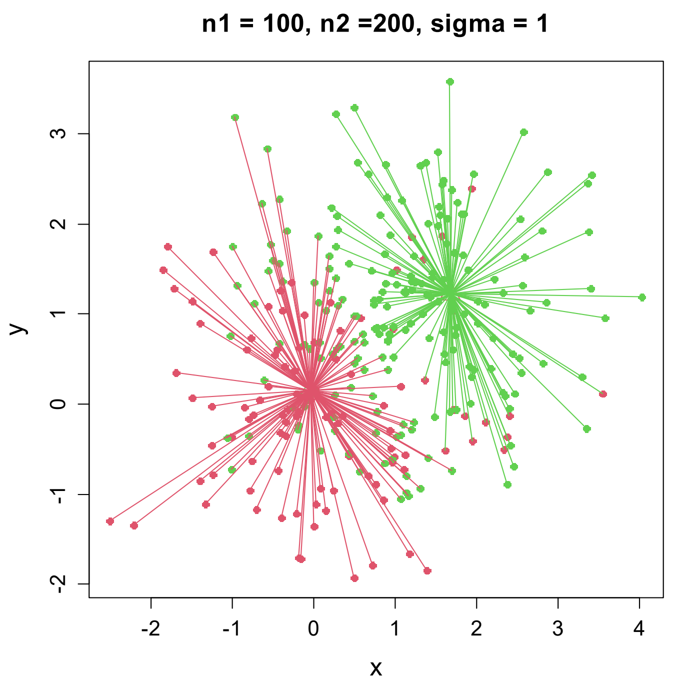
Output of `kmeans`:

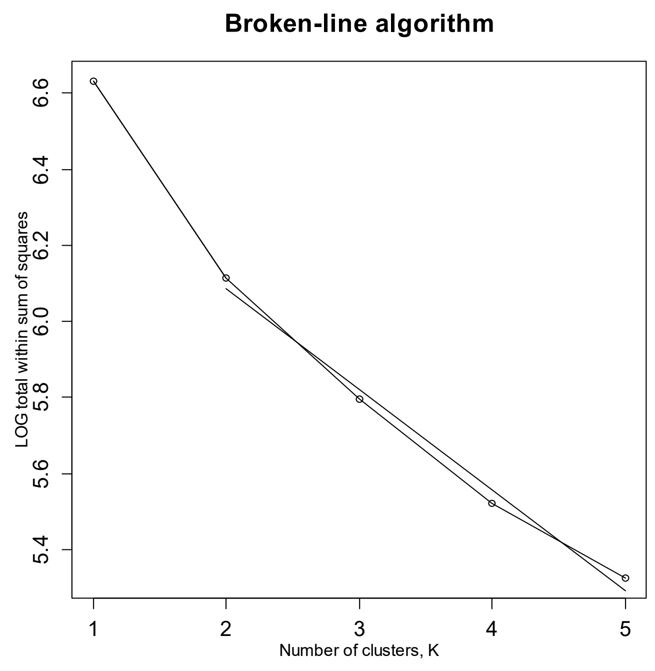
$$\begin{aligned}
\$\text{totss} &= \sum_{k=1}^{K}\sum_{i\in C_k}\|\mathbf{x}_i-\overline{\mathbf{x}}\|^2 \\
\$\text{withinss} &= \left\{\sum_{i\in C_k}\|\mathbf{x}_i-\overline{\mathbf{x}}\|^2, k=1,2,...,K\right\} \\
\$\text{tot.withinss} &= \sum_{k=1}^{K}\sum_{i\in C_k}\|\mathbf{x}_i-\overline{\mathbf{x}}_k\|^2
\end{aligned}$$

Example 2 *Generate two clusters with fixed centers but different σ and apply the K-means algorithm.*

Solution.

```
 kmsim=function(n1=100,n2=200,sigma=3)
{
dump("kmsim","c:\\QBS124\\kmsim.r")
X1=matrix(rnorm(n1*2,sd=sigma),ncol=2)
X2=matrix(rnorm(n2*2,mean=1,sd=sigma),ncol=2)
X=rbind(X1,X2)
n=n1+n2
plot(X[,1],X[,2],type="n",xlab="x",ylab="y",main=paste("n1 = ",n1,", n2 =",n2,", sigma
=",sigma,sep=""))
points(X1[,1],X1[,2],col=2,pch=16)
points(X2[,1],X2[,2],col=3,pch=16)
ok=kmeans(X,centers=2)
print(ok)
points(ok$centers[1,1],ok$centers[1,2],col=2,pch=2,cex=2)
points(ok$centers[2,1],ok$centers[2,2],col=3,pch=2,cex=2)
id=ok$cluster
n1.cl=length(id[id==1])
n2.cl=length(id[id==2])
segments(X[id==1,1],X[id==1,2],rep(ok$centers[1,1],n1.cl),rep(ok$centers[1,2],n1.cl),col=3)
segments(X[id==2,1],X[id==2,2],rep(ok$centers[2,1],n2.cl),rep(ok$centers[2,2],n2.cl),col=2)
}
```





Model-based K-means algorithm via maximum likelihood

Task: divide n observations $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in R^m$ into K clusters (given).

Assumption of the K-means algorithm:

1. The distribution of observations from the same cluster is Gaussian, with the common mean and variance σ^2 (spherical Gaussian).
2. Different clusters differ by the means $\boldsymbol{\mu}_k$, $k = 1, 2, ..., K$.

Statistical model for the K-means algorithm:

$$\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma^2 \mathbf{I}_m), \quad i \in C_k.$$

The parameters to estimate are the means $\{\boldsymbol{\mu}_k, k = 1, 2, ..., K\}$ and the common variance σ^2, but most importantly the index sets $(C_1, C_2, ..., C_K)$. The twice negative log-likelihood function takes the form

$$-l(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, C_1, ..., C_K) = mn \ln \sigma^2 + \sigma^{-2} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 .$$

Differentiating with respect to $\boldsymbol{\mu}_k$, we find that, given the index sets, the maximum likelihood (ML) estimator is

$$\overline{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{x}_i,$$

where n_k is the number of elements in cluster k. Differentiating l with respect to σ^2, we find that the ML estimation, i.e. maximization of function l, is equivalent to the minimum of the **total within sum of squares** (`$tot.withinss`),

$$S_K = \min_{C_1, ..., C_K} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2 .$$

Thus, ML with a spherical Gaussian distribution is equivalent to the traditional K-means algorithm. The minimization of criterion is not trivial and may have multiple minima, so several starting points may be used to confirm that the global minimum is found.

An ML estimate of the variance is

$$\widehat{\sigma}^2 = \frac{1}{nm} S_K,$$

Implications:

- The K-means algorithm is only applicable to normally distributed data with equal variance. Consequently, the K-means algorithm is not justified for uniformly distributed data or when vector components are measured on different scales and therefore have different variances.
- The K-means algorithm requires the same variance for all components. Usually we normalize the original data by subtracting the gross mean and dividing by the standard deviation (remove the scale difference) but such normalization would be suboptimal because the variance should be computed around the mean in each cluster, not around the gross mean.

Testing for the presence of clusters

A fundamental question is: are there clusters? A false clusterization. is illustrated in Figure 2. The K-means algorithm with two clusters ($K = 2$) is applied to $n = 100$ points generated from the same normal distribution with zero mean, unit variance, and zero correlation (spherical Gaussian distribution). The K-means algorithm divides these points into two clusters, but in fact there are no clusters because points are generated from the same distribution. Visualization may be deceiving. Needless to say, absence of clusters becomes even more difficult to detect for higher dimensions ($m > 2$).

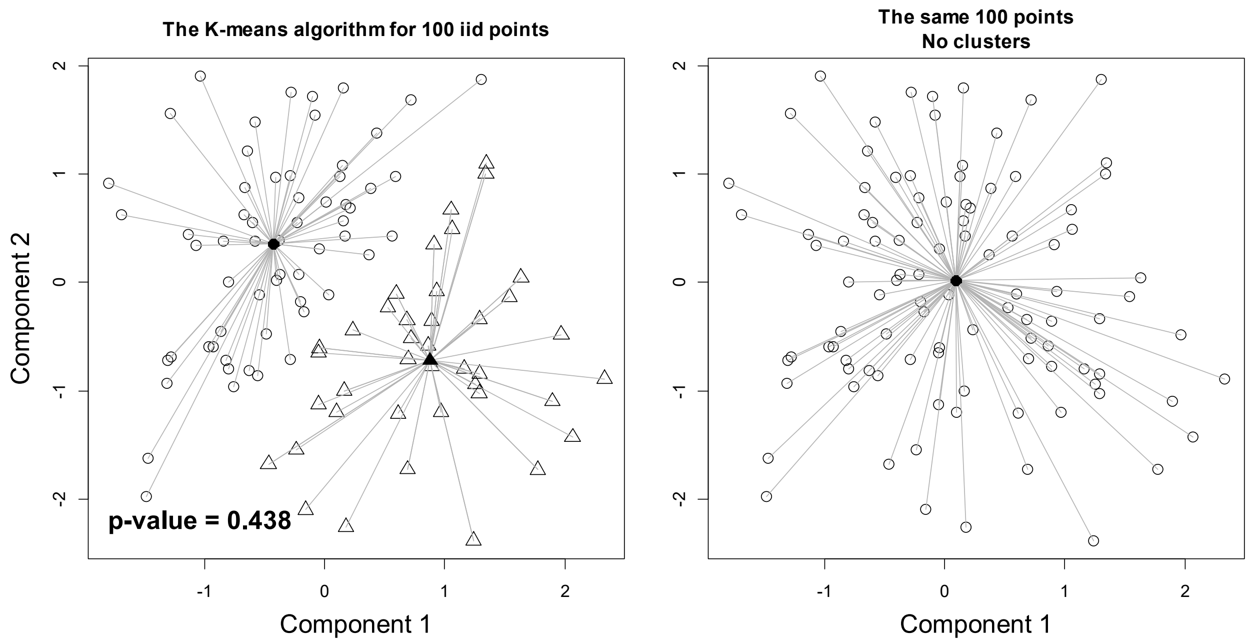


Figure 2: *The K-means algorithm with $K = 2$ for a sample of 100 random points from the same bivariate normal distribution with zero mean and unit variance. A wrong clusterization is shown in the right plot (the same points)!*

We aim to test if points $\{\mathbf{x}_i, i = 1, 2, ..., n\}$ belong to the same normal population —i.e., there are no clusters. This hypothesis will be referred to as the *no-clusters* hypothesis. The key observation is that, for the K-means algorithm, the index sets are unknown and subject to estimation. Therefore, a distribution, such as the F-distribution, does not hold. This distribution will be derived via simulations.

We say that there are no clusters if the null hypothesis $H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = ... = \boldsymbol{\mu}_K$ is not rejected with the given Type I error α (typically, $\alpha = 0.05$). If the index sets, C_k, were known, the traditional exact F-test or approximate likelihood ratio (LR) MANOVA test could be applied. These are based on the total and within-cluster sums of squares,

$$S_1 = \sum_{i=1}^{n} \|\mathbf{x}_i - \overline{\mathbf{x}}\|^2, \quad S_K = \min_{C_1,...,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2, \tag{1}$$

respectively. When the index sets are unknown and estimated, as in the K-means algorithm, the distribution of classical statistics does not hold, so the classical MANOVA does not apply.

To compute the p-value for the no-clusters hypothesis when the index sets are unknown, we need to estimate the cumulative distribution function (cdf) of statistics under the null hypothesis: i.e., when $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$, $i = 1, 2, ..., n$. We could use either the F-statistic, $(S_1 - S_K)/S_K$, or the likelihood ratio test, $\log(S_1/S_K)$, but the p-value does not change upon any strictly increasing transformation, so it suffices to find the cdf of the ratio,

$$r = \frac{S_1}{S_K}. \tag{2}$$

The advantage of statistic (2) is that its distribution, under the null hypothesis, does not depend on $\boldsymbol{\mu}$ and σ^2. Indeed, simple algebra proves that

$$r = \frac{S_1/\sigma^2}{S_K/\sigma^2} = \frac{S_{1z}}{S_{Kz}},$$

where

$$S_{1z} = \sum_{i=1}^{n} \|\mathbf{z}_i - \overline{\mathbf{z}}\|^2, \quad S_{Kz} = \min_{C_1,...,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{z}_i - \overline{\mathbf{z}}_k\|^2$$

and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

p-value for no-clusters hypothesis

The method of computing the p-value for the no-clusters null hypothesis versus the alternative that the number of clusters is K is as follows:

Let the K-means algorithm for the data at hand $\{\mathbf{x}_i, i = 1, 2, ..., n\}$ produce r_* as the ratio of two sums of squares (2).

Carry out a fairly large number of simulations N, say, $N = 1,000$, to obtain the empirical cdf of r: For each simulation,

1. generate $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$,
2. run K-means,
3. compute the total sum of squares S_{1z}, the within sum of squares from the K-means, S_{Kz}, and $r = S_{1z}/S_{Kz}$.

Plot the empirical cdf of r at the end.

Then, the p-value is the proportion of simulations in which $r > r_*$ where r_* is the observed value.

If there were clusters, then r_* would be greater than the typical r under the null hypothesis (no clusters). Typically, we say that the null hypothesis is rejected if the proportion (p-value) < 0.05.

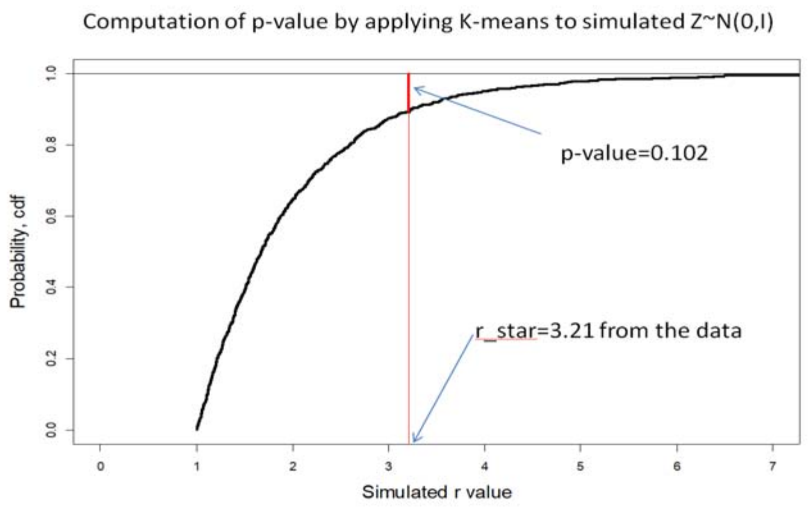
Null hypothesis: number of clusters=1, Alternative hypothesis: number of clusters=K

Proposition 1. *Let* $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma^2 \mathbf{I}_m)$, $\quad i \in C_k$ *as in K-means model. Then*

$$r = \frac{\textit{Total SS}}{\textit{Within SS}} = \frac{S_{1z}}{S_{Kz}} = \frac{\sum_{i=1}^{n} \|\mathbf{z}_i - \overline{\mathbf{z}}\|^2}{\min_{C_1,\dots,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{z}_i - \overline{\mathbf{z}}_k\|^2}$$

where $\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{I}_m)$. *Under the null hypothesis (single cluster)* $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \dots = \boldsymbol{\mu}_K = \mathbf{0}$

P-value computation



The null hypothesis: $H_0 : K = 1$, that is, no clusters.

The alternative hypothsis: H_A : there are $K > 1$ clusters.

The p-value for the configuration of points depicted in Figure 2 is 0.438. This means that the no-clusters hypothesis cannot be rejected.

Variance explained by cluster analysis

Draw the connection to the coefficient of determination in linear regression (the variance explained by predictors), a goodness of fit,

$$R^2 = 1 - \frac{\sum r_i^2}{\sum (y_i - \overline{y})^2} = 1 - \frac{\text{Unexplained SS}}{\text{Total SS}}.$$

Use the SS decomposition:

$$\text{Total SS} = \text{Within SS} + \text{Between SS}$$

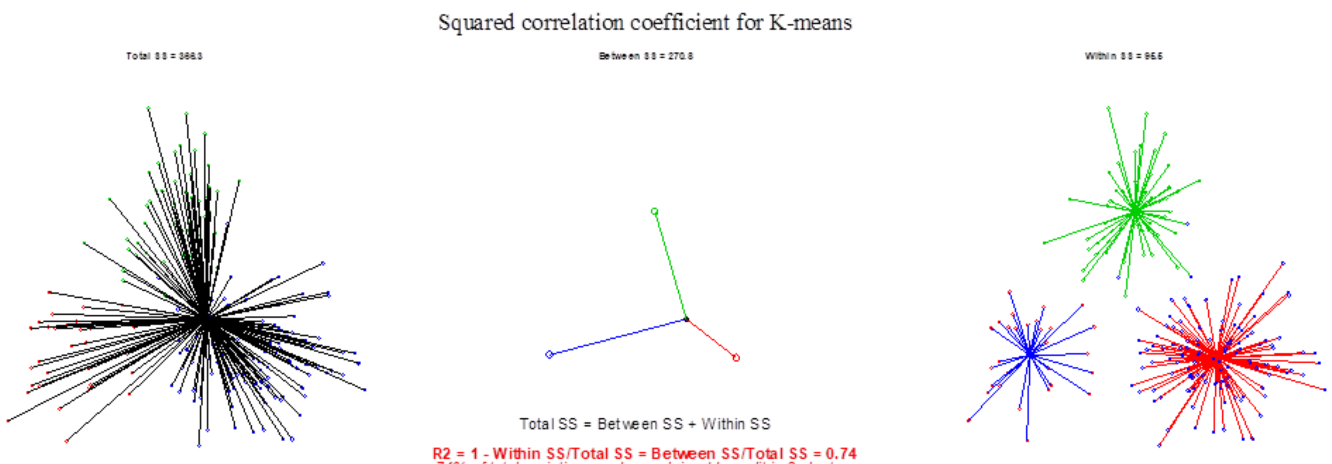
or

$$\sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}\|^2 = \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \overline{\mathbf{x}}_k\|^2 + \sum_{k=1}^{K} n_k \|\overline{\mathbf{x}}_k - \overline{\mathbf{x}}\|^2 .$$

Define

$$\begin{aligned} \text{Variance explained by clustering} &= 1 - \frac{1}{r} = 1 - \frac{\text{Within SS}}{\text{Total SS}} \\ &= 1 - \frac{\text{Unexplained SS}}{\text{Total SS}}. \end{aligned}$$

K-means variance decomposition and coefficient of determination/squared correlation coefficient



Total SS = Between SS + Within SS

R2 = 1 - Within SS/Total SS = Between SS/Total SS = 0.74
74% of total variation can be explained by split in 3 clusters

Computation the probability that vector belongs to cluster

Problem 1: compute the probability that a new vector $\mathbf{x}$ belongs to cluster k.

Solution. Since $\mathbf{x} \in R^m$ we have

$$\frac{1}{\sigma^2}\left\|\mathbf{x}-\overline{\mathbf{x}}_k\right\|^2 \sim \chi^2(m)$$

The probability that a new vector $\mathbf{x}$ belongs to cluster k is measured as the chi-square probability beyond $\mathbf{x}$:

$$\Pr(\mathbf{x}\in C_k) = 1 \text{ - pchisq(} \left\|\mathbf{x}-\overline{\mathbf{x}}_k\right\|^2/\widehat{\sigma}^2, \text{df=m)=pchisq(} \left\|\mathbf{x}-\overline{\mathbf{x}}_k\right\|^2/\widehat{\sigma}^2, \text{df=m,lower.tail=F)}$$

where $\widehat{\sigma}^2$ is the ML estimator, i.e.

$$\widehat{\sigma}^2 = \frac{1}{mn}S_K$$

where S_K is the total within sum of squares.

Another approach is to rerun `kmeans` with $\mathbf{x}$ attached to the previous data.

Problem 2: how to draw the 95% confidence circle around any cluster on the plane?

Solution. Use the fact that if the points $\mathbf{x} \in R^m$ then

$$\frac{1}{\widehat{\sigma}^2}\left\|\mathbf{x}-\overline{\mathbf{x}}_k\right\|^2 \sim \chi^2(m)$$

and therefore the radius of the circle/sphere

$$r = \widehat{\sigma}\sqrt{\chi^{-2}(1-\alpha, m)}$$

where $\chi^{-2}(1-\alpha, m)$ is the quantile of the chi-distribution with $\alpha = 0.05$.

How many clusters: the broken-line algorithm

"What is K?" is the paramount question of cluster analysis.

Our broken-line algorithm is an elaboration of the well-known and loosely defined *elbow method*:

(1) Plot the **log** total within sum of squares, S_K, against K for a sequence of values $K = 1, 2, ..., K_{\max}$.

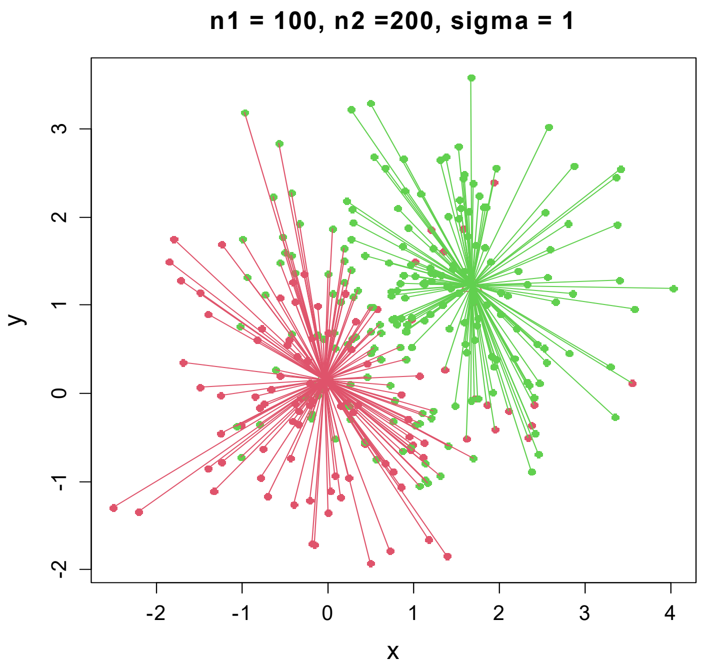
(2) Chose K at the elbow of the curve, i.e. where the line exhibits a change of slope.

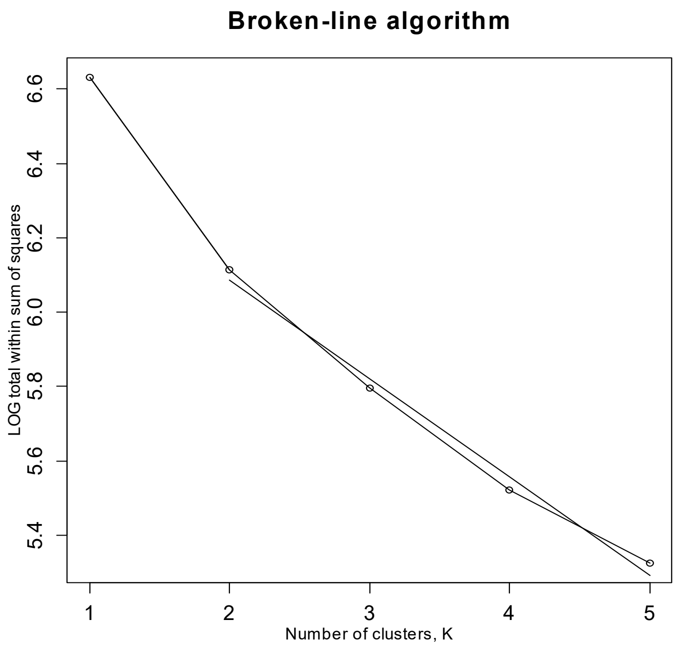
We facilitate the determination of K by plotting $\ln S_K$ and identifying K where the rate of decrease of $\ln S_K$ (the slope) changes. Precisely, the broken-line algorithm is as follows: Fit two linear regressions using two segments of the data, $\{S_1, S_2, .., S_K\}$ and $\{S_{K+1}, S_{K+2}, .., S_{K_{\max}}\}$ and compute the total residual sum of squares for $K = 2, 3, ..., K_{\max} - 2$. The optimal K is where the sum of squares takes a minimum.

Generic algorithm for identification of K:

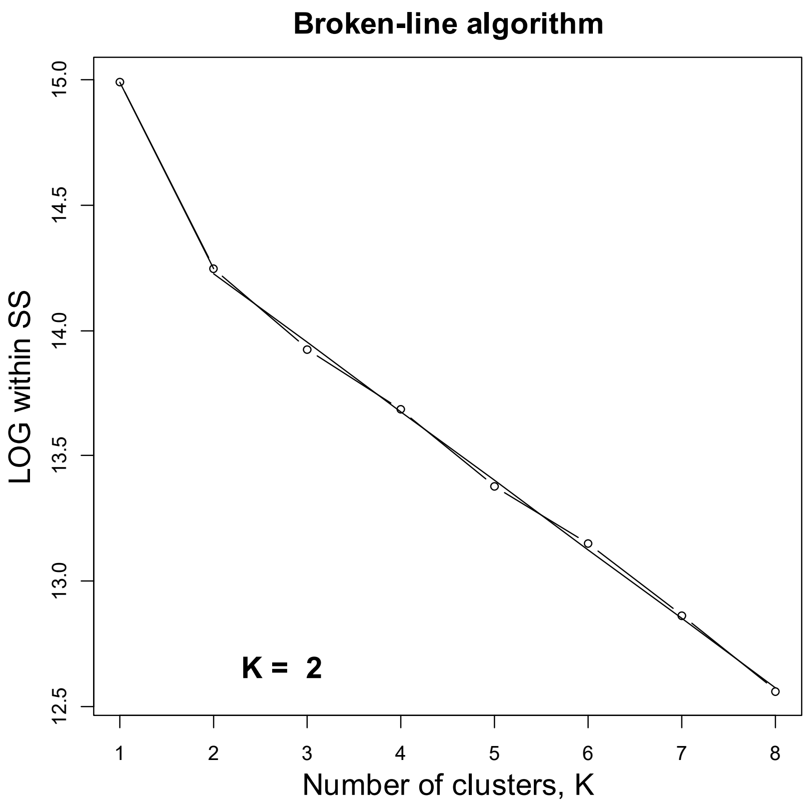
```
Kmax=15 #user-defined
ss=rep(NA,Kmax)
for(i in 1:Kmax)
    ss[i]=kmeans(da,centers=i,nstart=10)$tot.withinss
ss=log(ss)
plot(1:Kmax,ss,type="o",xlab="",ylab="",main="Broken-line algorithm")
mtext(side=1,"Number of clusters, K",cex=1,line=2)
mtext(side=2,"LOG total within sum of squares",cex=1,line=2)

siMIN=10^20
for(ik in 2:(Kmax-2))
{
    x1=1:ik;y1=ss[x1];x2=ik:Kmax;y2=ss[x2]
    si=sum(lm(y1~x1)$residuals^2)+sum(lm(y2~x2)$residuals^2)
    if(si<siMIN)
    {
        siMIN=si
        km=ik
    }
}
x1=1:km;y1=ss[x1];x2=km:Kmax;y2=ss[x2]
y1=lm(y1~x1)$fitted.values;lines(x1,y1)
y2=lm(y2~x2)$fitted.values;lines(x2,y2)
segments(km,-1,km,ss[km])
print(paste("Optima K =",km))
return(km)
```

See `kmsim.r` and `kmsimK.r`



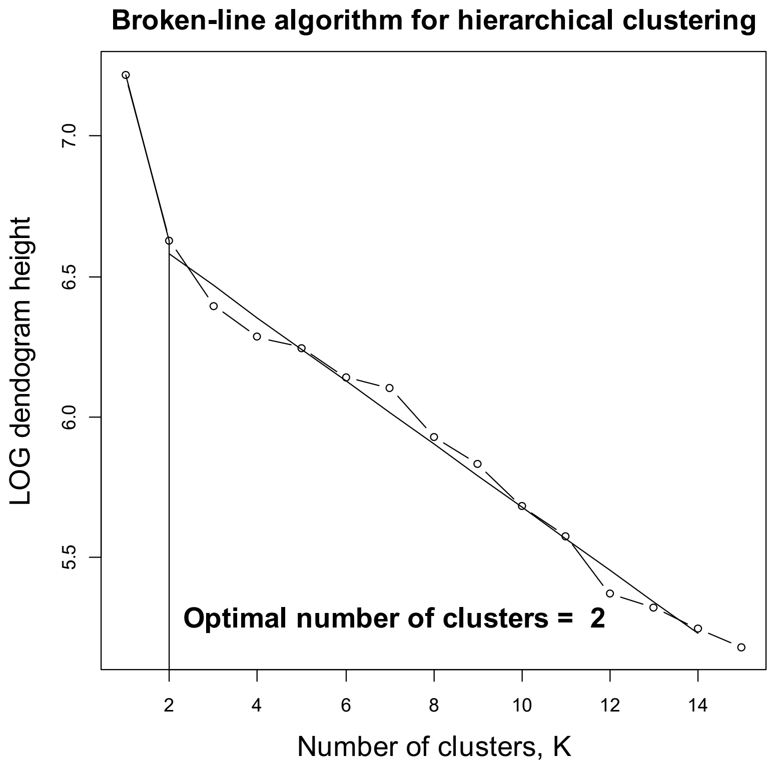


Application to city crime classification
`crime(job=2)`



Application of the broken-line algorithm to hiererachical clustering

Use LOG height instead of LOG within SS. See the R code `crime(job=1.1)`



Quiz
1. Does cluster analysis belong to a supervised learning? **No**
2. Does discriminant analysis belong to a supervised learning? **Yes**
3. Does logistic regression belong to a supervised learning? **Yes**
4. Does ROC curve belong to a supervised learning? **Yes**
5. Can ROC curve be applied to cluster analysis data? **No**
6. Can ROC curve be applied to discriminant analysis data? **Yes**
7. Can ROC curve be applied to logistic regression data? **Yes**