Lecture 6: Clustering

Jing Ma, Statistics, TAMU

14 October, 2019

Recap

- Mixture models (finite and infinite)
- EM algorithm
- Variance-stabilizing transformation

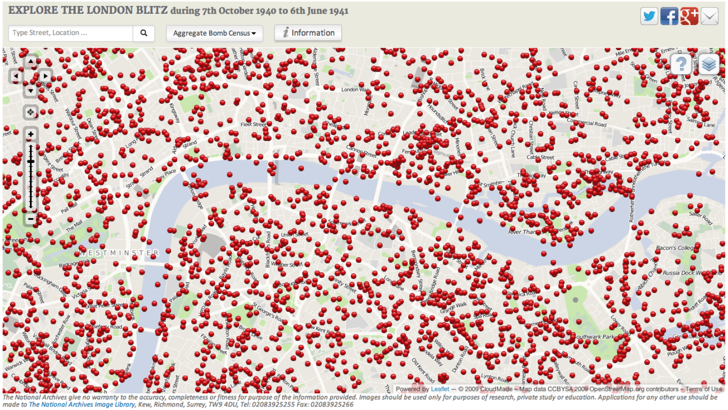
This lecture

- See **distances** that help us define clusters.
- Learn *k*-means and hierarchical clustering.
- How many clusters are there?

Clustering

Finding a latent or hidden variable present which we are not necessarily provided.

London bombings

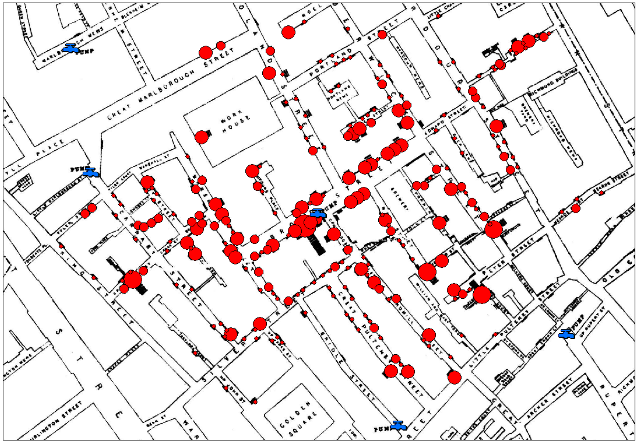

EXPLORE THE LONDON BLITZ during 7th October 1940 to 6th June 1941

The National Archives give no warranty to the accuracy, completeness or fitness for purpose of the information provided. Images should be used only for purposes of research, private study or education. Applications for any other use should be made to The National Archives Image Library, Kew, Richmond, Surrey, TW9 4DU, Tel: 02083925255 Fax: 02083925266

- The red dots designate locations that were bombed during World War II.
- Many attempts to find a rational explanation for the bombing patterns.

Cholera map in 1850s

John Snow and the Broad Street Pump



Breast cancer incidence rates vary geographically

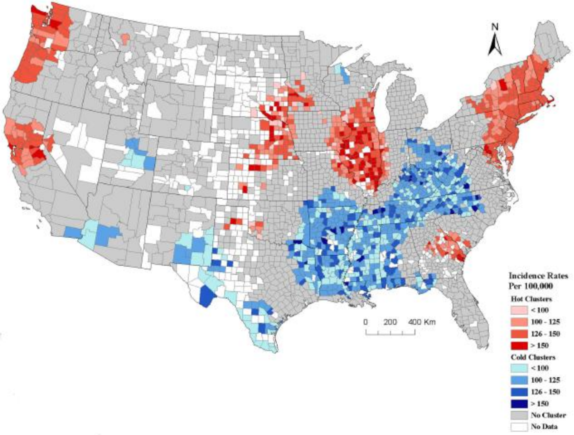
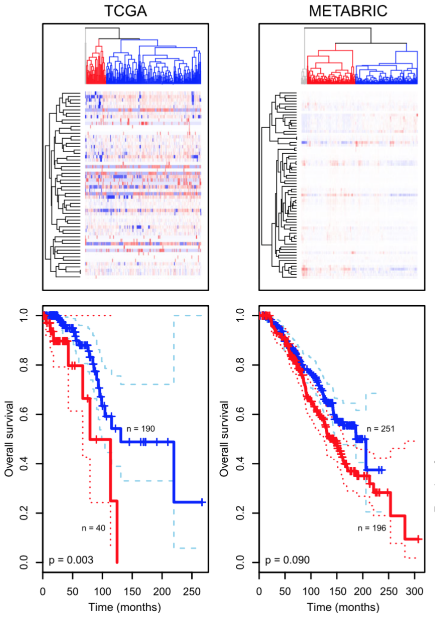


Figure 2
Geographical clusters of U.S. counties with significant high or low breast cancer incidence rates among Caucasians (including Hispanics) analyzed at a 200 km distance band. The graduated red and blue colors show high (hot) and low (cold) clusters respectively, for age-adjusted average annual incidence rates (2000/2001-2004/2005) of breast cancer. The counties with no color either have no data or counts less than 3-5. Graduated colors were assigned to the hot and cold clusters based on the incidence rate of individual counties. In total there were 2,692 counties used in the breast cancer cluster analysis. The inserted regional U.S. map depicts the regions of the U.S. used to describe the cluster patterns. Data source: National Cancer Institute-State Cancer Profiles and State Cancer Registries.

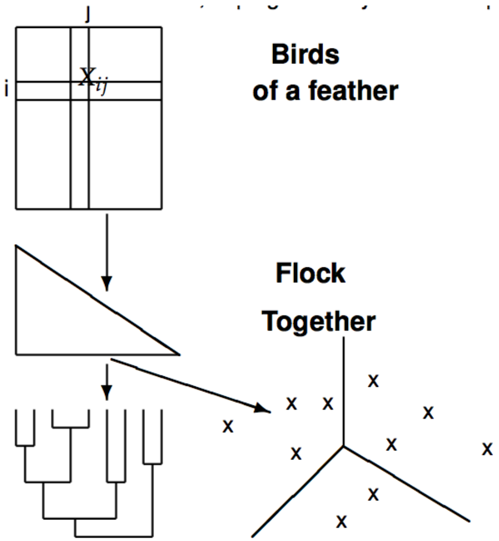
Breast cancer clusters by molecular signatures



Clustering algorithms

- There are already thousands of existing clustering algorithms combining different choices for the basic distance used to compare observations, ways of combining similar observations and ways of deciding how many cluster exist.
- We will learn the basics of how clustering algorithms are designed.

Clustering workflow



Clustering workflow

- Start from an observation by feature matrix X.
- Choose a distance matrix.
- Given the distance matrix, we can group the observations either by building a hierarchical clustering tree or using partitioning methods.

Both types of methods require a choice to be made: the number of clusters.

- For partitionning approaches such as k-means, this choice has to be made at the beginning.
- For hierarchical clustering this can be deferred to the end of the analysis.

How do we measure similarity?

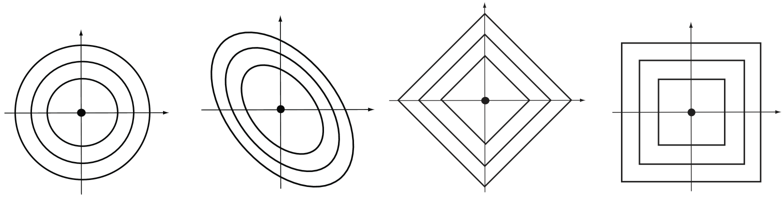


Fig. 1: Equal-distance contour plots according to four different distances: points on any one curve are all the same distance from the center point.

Euclidean distance

- Consider two points $A = (a_1, ..., a_p)$ and $B = (b_1, ..., b_p)$ in a p-dimensional space (for the p features)
- The Euclidean distance between A and B is the square root of the sum of squares of the differences in all p coordinate directions:

$$d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_p - b_p)^2}.$$

Weighted Euclidean distance

- A generalization of the ordinary Euclidean distance, by giving different directions in feature space different weights.
- e.g. Mahalanobis distance with center μ and covariance Σ:

$$D(x,\mu)^2 = (x-\mu)'\Sigma^{-1}(x-\mu)$$

Manhattan distance

- The Manhattan or L_1 distance takes the sum of the absolute differences in all coordinates.

$$d(A, B) = |a_1 - b_1| + |a_2 - b_2| + ... + |a_p - b_p|.$$

Maximum distance

- The maximum of the absolute differences between coordinates is also called the L_∞ distance:

$$d_\infty(A, B) = \max_i |a_i - b_i|.$$

Minkowski distance

- Allowing the exponent to be m instead of 2, as in the Euclidean distance, gives the Minkowski distance

$$d(A, B) = \{(a_1 - b_1)^m + (a_2 - b_2)^m + ... + (a_p - b_p)^m\}^{\frac{1}{m}} .$$

Jaccard Index

- Occurrence of traits or features in ecological or mutation data can be translated into presence and absence and encoded as 1's and 0's.
- In such situations, co-occurence is often more informative than co-absence. For instance, when comparing mutation patterns in HIV, the co-existence in two different strains of a mutation tends to be a more important observation than its co-absence.
- For this reason, biologists use the **Jaccard index**.

Jaccard distance

- Let's call our two observation vectors A and B, f_{11} the number of times a feature co-occurs in A and B, f_{10} (and f_{01}) the number of times a feature occurs in A but not in B (and vice versa), and f_{00} the number of times a feature is co-absent. The Jaccard index is

$$J(A, B) = \frac{f_{11}}{f_{01} + f_{10} + f_{11}},$$

  (i.e., it ignores f_{00}).

- **Jaccard dissimilarity** is

$$d_J(A, B) = 1 - J(A, B) = \frac{f_{01} + f_{10}}{f_{01} + f_{10} + f_{11}}.$$

Correlation based distance

$$d(A, B) = \sqrt{2(1 - \mathrm{cor}(A, B))}.$$

Question

Which of the two cluster centers in the Figure is the red point closest to?

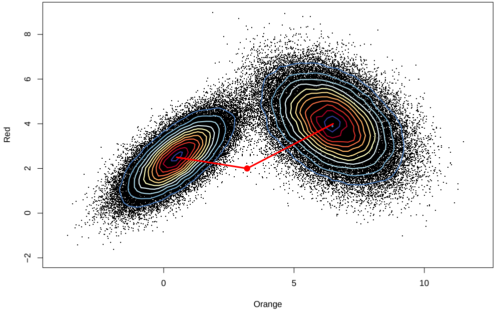


Fig. 2: An example for the use of Mahalanobis distances to measure the distance of a new data point (red) from two cluster centers.

```
## [1] 19.69269

## [1] 16.00794
```

Which distance to use?

There are almost a hundred different distances available combining outside information (distance on a graph, geodesic distance along a path, distance on a tree), combining categorical data and continuous data (Gower's distance) and using many different weighting schemes.

If you know what is the relevant notion of **closeness** or similarity for your data, you have (almost) solved the problem.

Computations related to distances in R

The `dist` function in **R**:

- is optimized to use less space than the full n^2 positions a complete $n \times n$ distance matrix between n objects would require.
- computes one of six choices of distance (*euclidean*, *maximum*, *manhattan*, *canberra*, *binary*, *minkowski*) and outputs a vector of values sufficient to reconstruct the complete distance matrix.
- returns a special object of class `dist` that encodes the relevant vector of size $n \times (n-1)/2$.

Computations related to distances in R

Here is the output for a 3 by 3 matrix:

```
mx  = c(0, 0, 0, 1, 1, 1)
my  = c(1, 0, 1, 1, 0, 1)
mz  = c(1, 1, 1, 0, 1, 1)
mat = rbind(mx, my, mz)
dist(mat)
```

```
##          mx       my
## my 1.732051
## mz 2.000000 1.732051
```

```
# dist(mat, method = "maximum")
```

Computations related to distances in R

In order to access a particular distance (for example the distance between observations 1 and 2), one has to turn the `dist` class object back into a matrix.

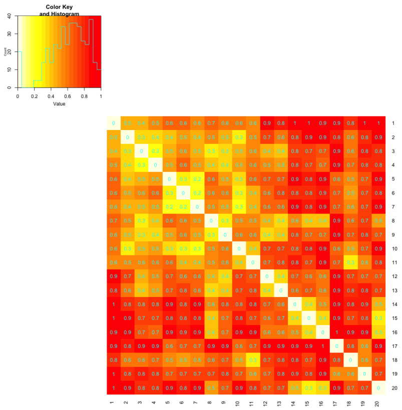
```r
as.matrix(dist(mat))[2, 1]
```

```
## [1] 1.732051
```

Computations related to distances in R

phyloseq distances

```r
require(vegan)
data(dune)
dist.dune=vegdist(dune)
# symnum(as.matrix(dist.dune))
```



Computations related to distances in R

- Let's look at how we would compute the Jaccard distance between HIV strains.
- Compare the Jaccard distance (available as the function `vegdist` in the R package **vegan**) between mutations in the HIV data `mut` to the correlation based distance.

```r
library("vegan")
mut = read.csv("data/HIVmutations.csv")
mutJ = vegan::vegdist(mut, "jaccard")
mutC = sqrt(2 * (1 - cor(t(mut))))
```

Hierarchical clustering

- When the number of clusters is not known a priori, it is useful to perform hierarchical clustering because it provides a graphical summary that enables us to evaluate where the cutoff for choosing clusters should occur.
- This class of methods starts with all the observations making their own cluster and agglomerates gradually the points together by similarity so the number of clusters decreases as points are clumped together into clusters, of course the user will have to make choices:
  - What is the relevant distance between observations to start with.
  - Which criteria should be used for grouping clusters that have been composed.
  - How should distances between these clusters be defined.

Hierarchical clustering

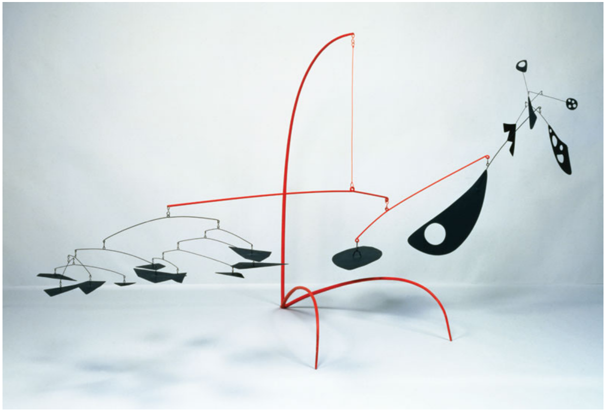


Fig. 3: Like a mobile

Computing dissimilarities between clusters

- **Minimal jump** (single linkage or nearest neighbor): This will separate the groups as much as possible,

$$D_{12} = min_{i \in C_1, i \in C_2} d_{ij}.$$

- This method tends to create clusters that look like contiguous strings of points. The cluster tree often looks like a **comb**.

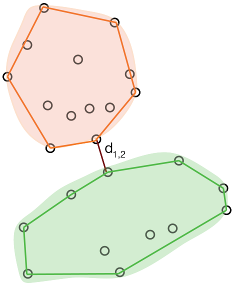


Fig. 4: Distance in the single linkage method.

Computing dissimilarities between clusters

- **Maximum jump** (complete linkage or furthest neighbor): The distances between clusters are determined by the greatest distance between any two objects in the different clusters. This method performs quite well in cases when the objects form naturally distinct `clumps`.

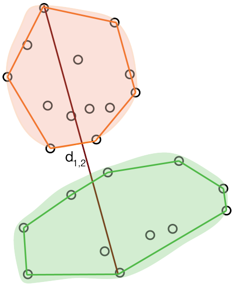


Fig. 5: Distance in the complete linkage method.

Computing dissimilarities between clusters

- **Mean**: Half way between the two above,

$$D_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, i \in C_2} d_{ij}.$$

  Unweighted pair-group average. (Goldilocks method)

- **Weighted pair-group average**.
- **Ward's method**:
  - An analysis of variance approach to evaluate the distances between clusters.
  - Minimize the Sum of Squares (SS) of any two (hypothetical) clusters that can be formed at each step.
  - This method is regarded as very efficient; however, it tends to create clusters of small size.

Interpretation of hierarchies

- **Symmetries**
- Ordering has to be interpreted carefully

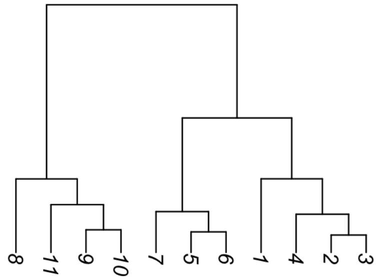


Fig. 6: This tree can be drawn in many different ways.

Advantages and Disadvantages of the various distances between clumps

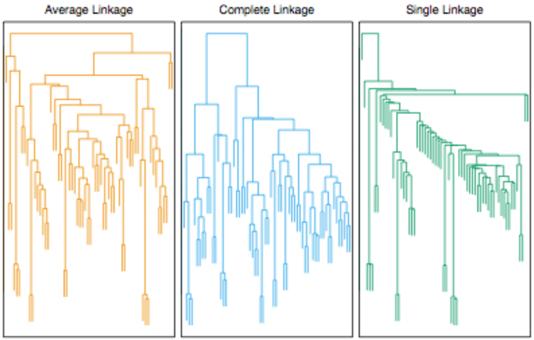
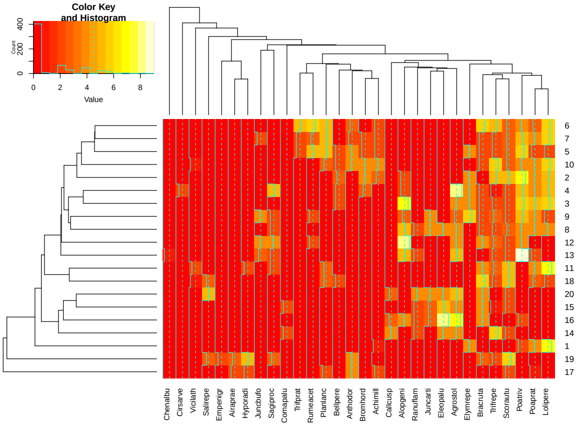


Fig. 7: Tree shapes differ.

Single linkage

Good for recognizing the number of clusters . . . . . . But combs.

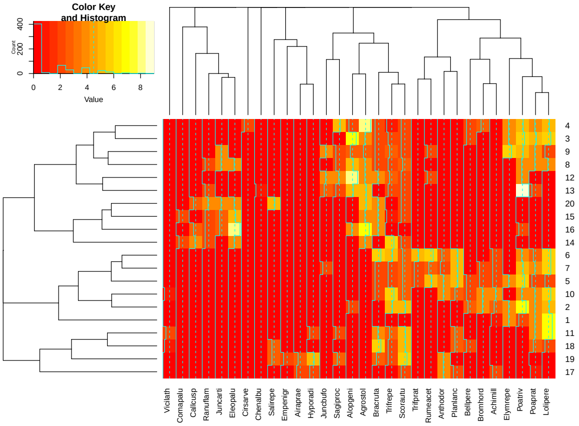
```
my.hclustfun=function(d){hclust(d, method = "single")}
my.distfun=function(x) {vegdist(x)}
heatmap.2(as.matrix(dune),hclustfun=my.hclustfun, distfun=my.distfun)
```



Maximal linkage

Compact classes ...... but one observation can alter groups.

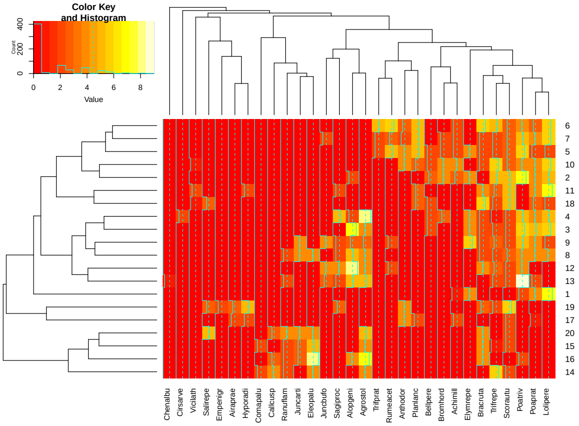
```
my.hclustfun=function(d){hclust(d, method = "complete")}
my.distfun=function(x) {vegdist(x)}
heatmap.2(as.matrix(dune),hclustfun=my.hclustfun,distfun=my.distfun)
```



Average linkage

Classes have similar size and variance.

```
my.hclustfun=function(d){hclust(d, method = "aver")}
my.distfun=function(x) {vegdist(x)}
heatmap.2(as.matrix(dune),hclustfun=my.hclustfun,distfun=my.distfun)
```

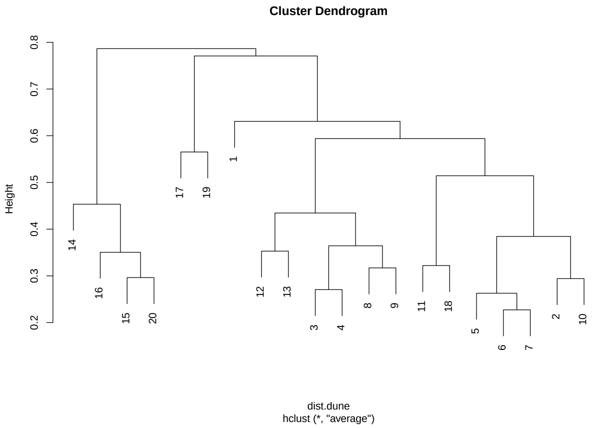


Ward's method

- Minimising an inertia (sums of squares of distances within groups).
- Classes are small if high variability observations.

Computing and interpreting hierarchical clustering trees

```
require(vegan)
hclust.aver <- hclust(dist.dune, method="aver")
plot(hclust.aver)
```



Nonparametric mixture detection

k-means and *k*-medoids

- Non-hierarchical clustering or iterative relocation methods.
- Several initial choices to be made with these methods and when the *a priori* knowledge is not available this can be a drawback.
  - The first is the number of clusters suspected.

Example of how kmeans works

The function kmeans is the one that can be used in **R**.

```
require(animation)
kmeans.ani(x = matrix(c(runif(100),runif(100)), ncol = 2,
                      dimnames = list(NULL, c("X1", "X2"))),
           centers = 3, pch = 1:3,col = 1:3,
           hints = c("Move centers", "Find clusters"))
```

Animated Gif

Example of how `kmeans` works

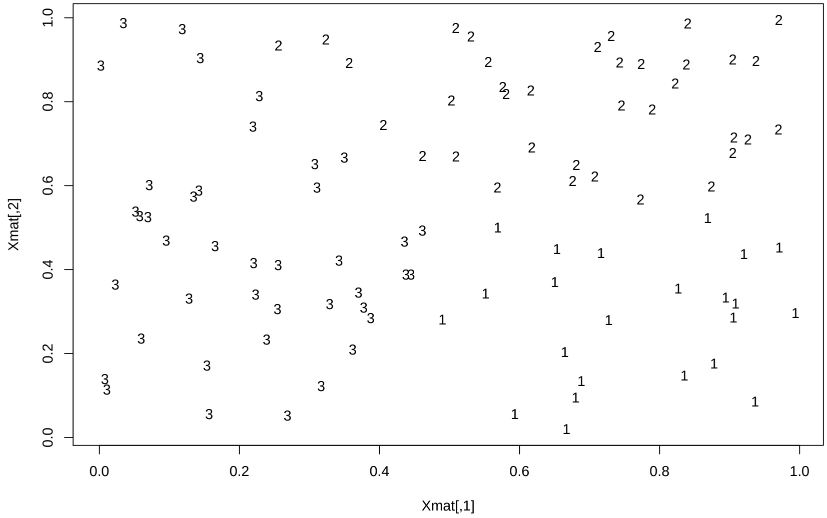
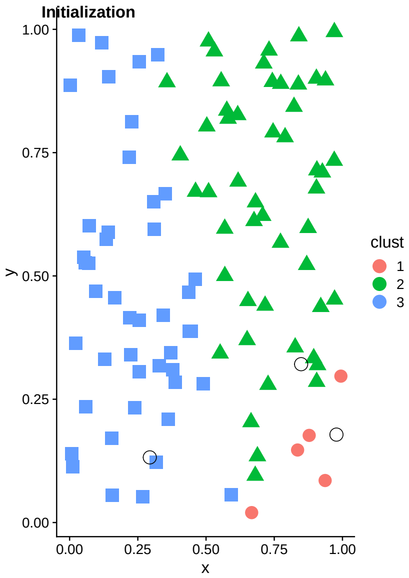
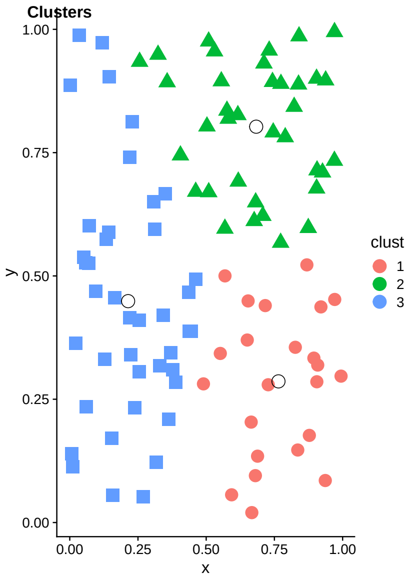


Fig. 8: The true cluster

Example of how `kmeans` works





k-medoids algorithm

1. For a given cluster assignment C, find the observation in the cluster minimizing total distance to other points in that cluster:

   $$i_k^* = \text{argmin}_{\{i:C(i)=k\}} \sum_{C(j)=k} D(x_i, x_j).$$

   Then $m_k = x_{i_k^*}, k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

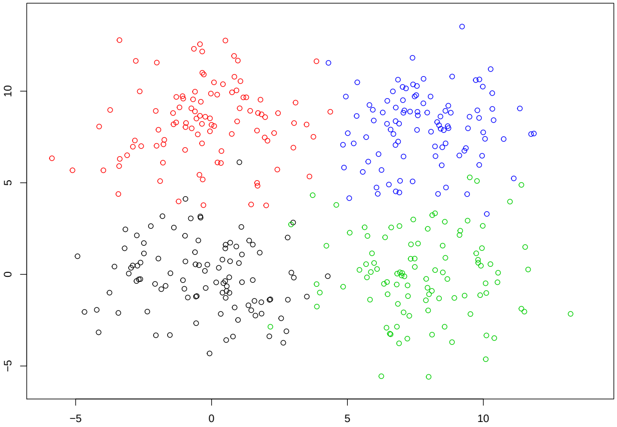
   $$C(i) = \text{argmin}_{1 \le k \le K} D(x_i, m_k).$$

Iterate steps 1 and 2 until the assignments do not change.

k-means vs k-medoids

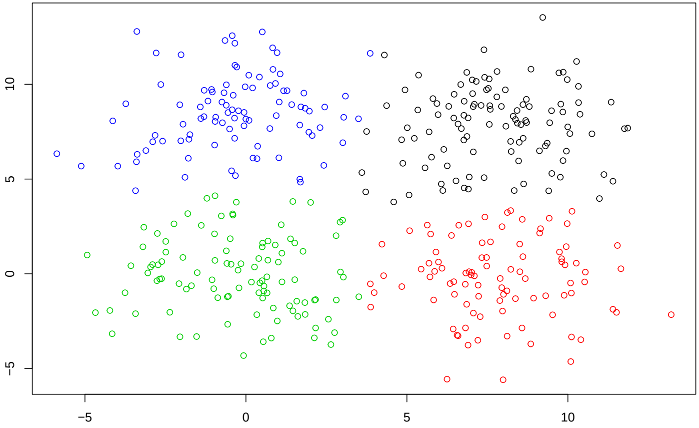
- k-means attempts to minimize the total squared error
- k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster.
- In contrast to the k-means algorithm which uses the means of the cluster as centers, k-medoids chooses data points as centers (medoids or exemplars).

Comparison of Truth, *k*-means and Hierarchical Clustering



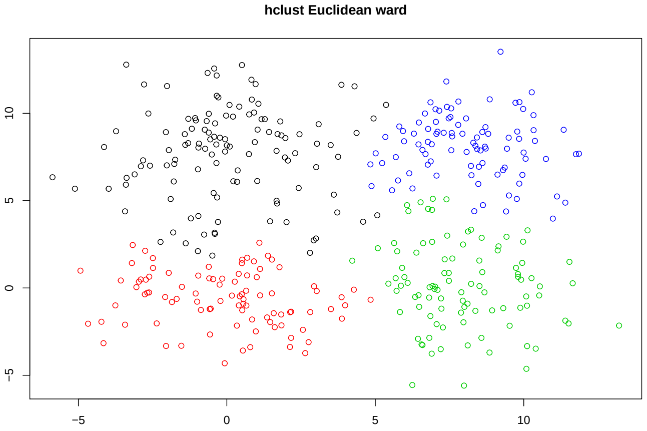
Compute kmeans classes with $k=4$

```r
km4 <- kmeans(simul4$values,4)
plot(simul4$values, col = km4$cluster,  xlab = "", ylab = "")
```



Plot the results of a hierarchical clustering assignment

```r
hc = hclust(dist(simul4$values), method = "ward.D")
memb <- cutree(hc, k = 4)
plot(simul4$values, col = memb, xlab = "", ylab = "",
     main="hclust Euclidean ward")
```



So far

- Distance measures
- Hierarchical clustering
- *k*-means clustering
- What remains: how to choose the number of clusters?

Option 1: gap statistic

Let's look at the gap statistic on a real data example.

```r
library("Hiiragi2013")
data("x")
```

We start by choosing the 50 most variable genes (features).

```r
# Define a function that takes the data and the number of clusters, and returns
pamfun = function(x, k)
  list(cluster = cluster::pam(x, k, cluster.only = TRUE))

selFeats = order(rowVars(Biobase::exprs(x)), decreasing = TRUE)[1:50]
embmat = t(Biobase::exprs(x)[selFeats, ])
embgap = cluster::clusGap(embmat, FUN = pamfun, K.max = 10,
                          verbose = FALSE)
```

Option 1: gap statistic

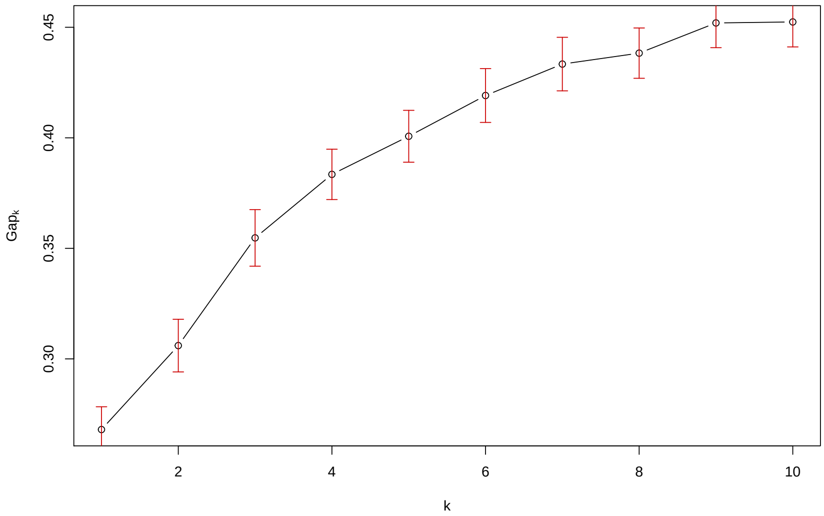


Fig. 9: The gap statistic for the Hiiragi2013 data

The number of clusters

```r
## Default method "firstSEmax"
k1 = maxSE(embgap$Tab[, "gap"], embgap$Tab[, "SE.sim"])

k2 = maxSE(embgap$Tab[, "gap"], embgap$Tab[, "SE.sim"],
           method = "Tibs2001SEmax")
c(k1, k2)
```

```
## [1] 9 7
```

- The default choice for the number of clusters, `k1`, is the first value of k for which the gap is not larger than the first local maximum minus a standard error s (see the manual page of the `clusGap` function). This gives a number of clusters $k = 9$.
- The choice recommended by Hastie et al. is the smallest k such that $\text{gap}(k) \geq \text{gap}(k+1) - s'_{k+1}$, which gives $k = 7$.

The number of clusters

```r
cl = pamfun(embmat, k = k1)$cluster
table(pData(x)[names(cl), "sampleGroup"], cl)
```

```
##                    cl
##                      1  2  3  4  5  6  7  8  9
##   E3.25             23 11  1  1  0  0  0  0  0
##   E3.25 (FGF4-KO)    0  0  1 16  0  0  0  0  0
##   E3.5 (EPI)         2  1  0  0  0  8  0  0  0
##   E3.5 (FGF4-KO)     0  0  8  0  0  0  0  0  0
##   E3.5 (PE)          0  0  0  0  9  2  0  0  0
##   E4.5 (EPI)         0  0  0  0  0  0  0  4  0
##   E4.5 (FGF4-KO)     0  0  0  0  0  0  0  0 10
##   E4.5 (PE)          0  0  0  0  0  0  4  0  0
```

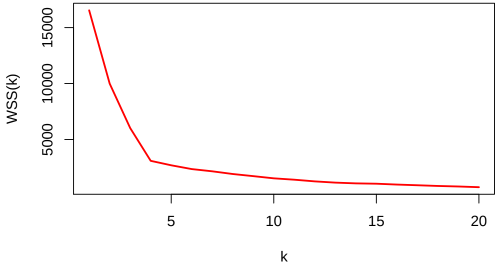
Option 2: elbows in the WSS plot

We can compute the kmeans within-groups sum of squared distances (WSS) for varying k.

```
wss = sum(apply(scale(simul4$value,scale=FALSE),2,function(x){x^2}))
for (k in 2:20) {
  km4 <- kmeans(simul4$values,k)
  wss <- c(wss, sum(km4$withinss))
}
```

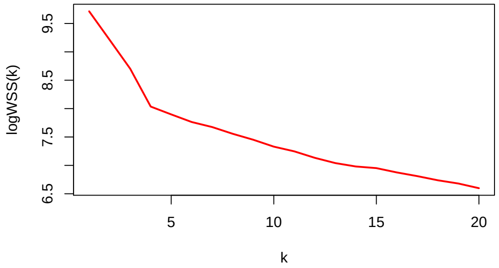
Option 2: elbows in the WSS plot

```
plot(1:20, wss, xlab='k', ylab='WSS(k)', type='l', lwd=2, col='red')
```



Option 2: elbows in the WSS plot

```r
plot(1:20, log(wss), xlab='k', ylab='logWSS(k)', type='l', lwd=2, col='red')
```



Option 3: silhouette plots

For each observation i, the silhouette width $s(i)$ is defined as follows:

- Let $a(i)$ be the average dissimilarity between i and all other points of the cluster to which i belongs (if i is the only observation in its cluster, $s(i) := 0$ without further calculations).
- For all other clusters C, put $d(i, C) =$ average dissimilarity of i to all observations of C.
- The smallest of these $d(i, C)$ is called

  $$b(i) := min_C d(i, C)$$

  and can be seen as the dissimilarity between i and its *neighbor* cluster, i.e., the nearest one to which it does not belong.
- Finally,

  $$s(i) := \frac{b(i) - a(i)}{\max(a(i), b(i))}.$$

Silhouette plots for simulated data

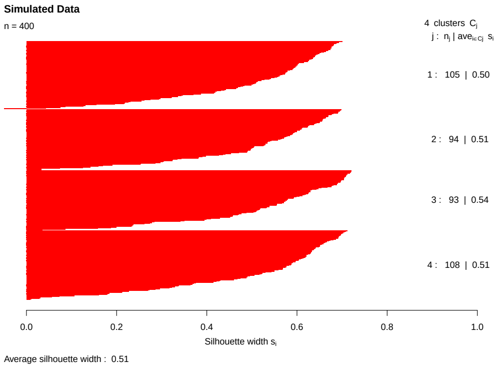
```r
library("fpc")

asw <- numeric(20)
for (k in 2:20)
  asw[[k]] <- cluster::pam(simul4$values, k)$silinfo$avg.width
k.best <- which.max(asw)
#cat("Silhouette-optimal number of clusters:", k.best, "\n")
k.best
```

```
## [1] 4
```

Silhouette plots for simulated data

```
p4=pam(x=simul4$values,k=k.best)
si=silhouette(p4)
plot(si,col="red",main="Simulated Data")
```



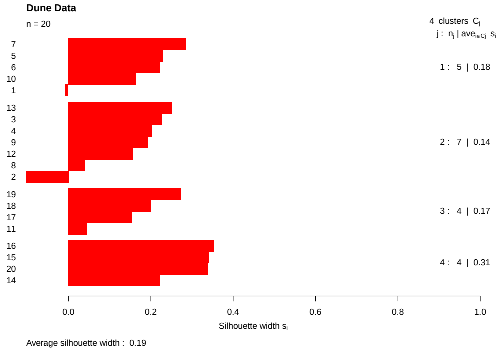
Silhouette plots for the Dune data

```r
asw <- numeric(20)
for (k in 2:18)
  asw[[k]] <- cluster::pam(dune, k)$silinfo$avg.width
k.best <- which.max(asw)
cat("Silhouette-optimal number of clusters:", k.best, "\n")
```

```
## Silhouette-optimal number of clusters: 4
```

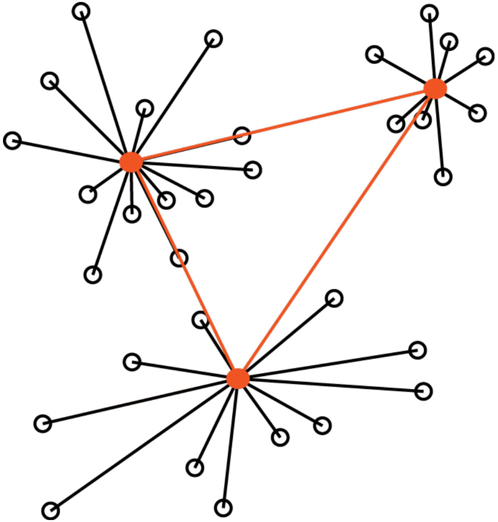
Silhouette plots for the Dune data

```
p4=cluster::pam(x=dune,k=k.best)
si=silhouette(p4)
plot(si,col="red",main="Dune Data")
```



Option 4: Calinski-Harabasz variance ratio

Within (black) and between (red) sums of squares:

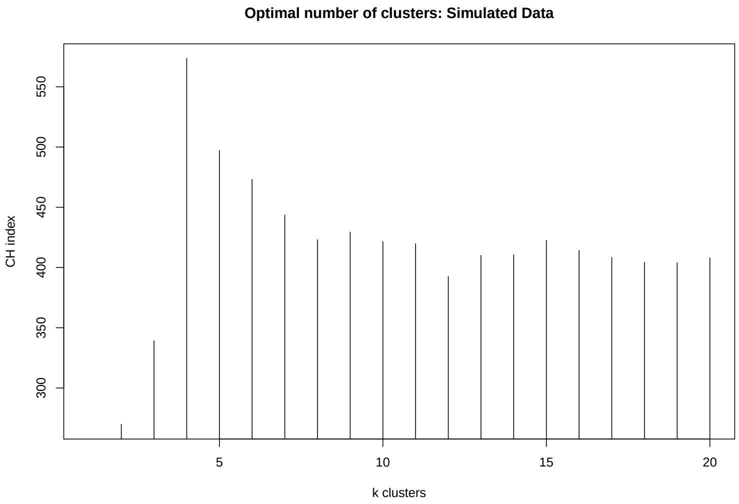


Calinski-Harabasz index on simulated data

```r
pam.clustering = function(x, k) {
  # x is a distance matrix and k the number of clusters
  cluster = as.vector(cluster::pam(as.dist(x), k, diss = TRUE)$clustering)
  return(cluster)
}
data = simul4$values
data.dist = dist(simul4$values)
data.cluster = pam.clustering(data.dist, k = 3)
require(clusterSim)

ch.index = NULL
for (k in 1:20) {
  if (k == 1) {
    ch.index[k] = NA
  } else {
    data.cluster_temp = pam.clustering(data.dist, k)
    ch.index[k] = index.G1(data, data.cluster_temp, d = data.dist,
                           centrotypes = "medoids")
  }
}
```

Calinski-Harabasz index on simulated data

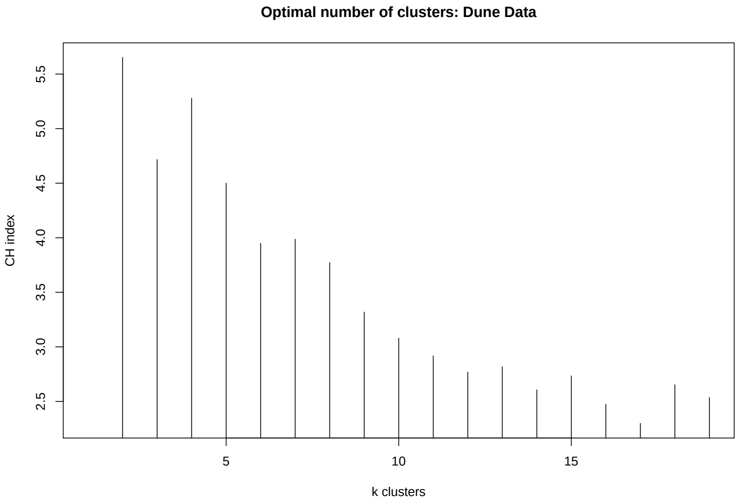


Calinski-Harabasz index on Dune data

```r
data = dune
data.dist = dist.dune
data.cluster = pam.clustering(data.dist, k = 3)
require(clusterSim)

ch.index = NULL
for (k in 1:19) {
  if (k == 1) {
    ch.index[k] = NA
  } else {
    data.cluster_temp = pam.clustering(data.dist, k)
    ch.index[k] = index.G1(data, data.cluster_temp, d = data.dist,
                           centrotypes = "medoids")
  }
}
```

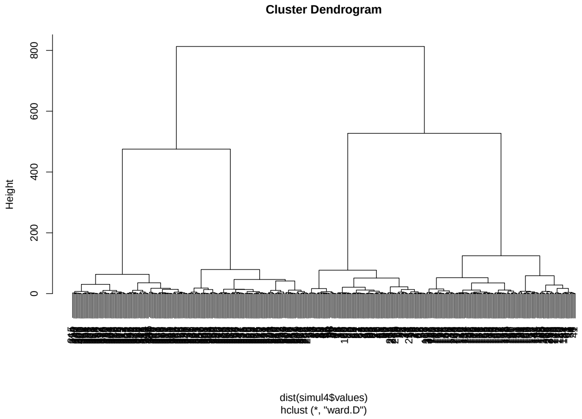
Calinski-Harabasz index on Dune data



Option 5: visual assessment

Look at the long edges in hierarchical clustering

```
plot(hclust(dist(simul4$values), method = "ward.D"))
```



Choosing the number of clusters

Detailed graphical account on stackoverflow

Gene expression clustering

```
load("data/Msig3transp.RData")
dim(Msig3transp)
```

```
## [1]  30 156
```

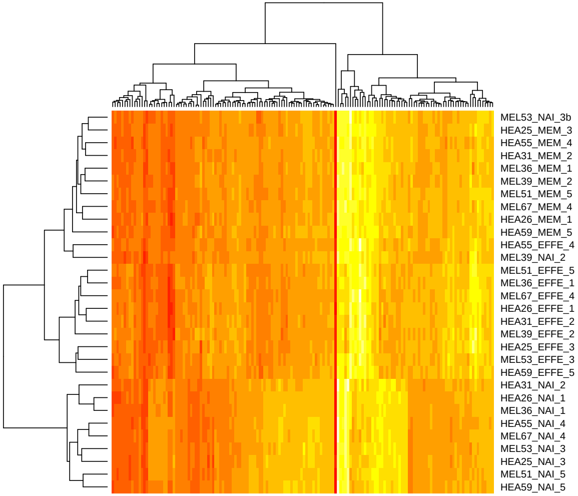
```
head(Msig3transp[,1:3])
```

```
##                   X3968     X14831      X13492
## HEA26_EFFE_1 -2.6108361 -1.1857923 -0.05612926
## HEA26_MEM_1  -2.2592865 -0.4718562  0.27730991
## HEA26_NAI_1  -0.2719081  0.8170308  0.81269167
## MEL36_EFFE_1 -2.2431022 -1.0806009 -0.23557156
## MEL36_MEM_1  -2.6798807 -0.1463671  0.24909402
## MEL36_NAI_1  -0.4685057  1.0693272  0.75844932
```

```
# dr=round(dist(Msig3transp),2)
```

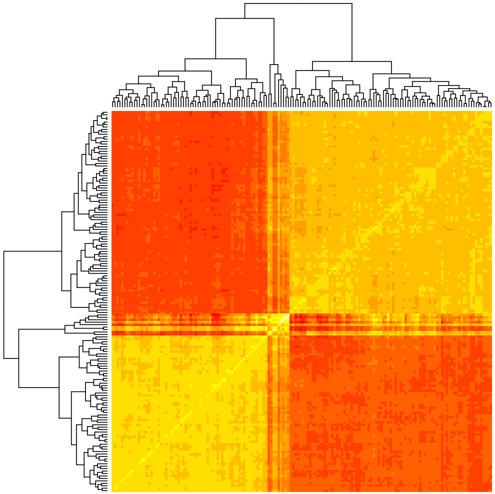
Heatmap of data

```r
heatmap(as.matrix(Msig3transp),labCol = "")
```



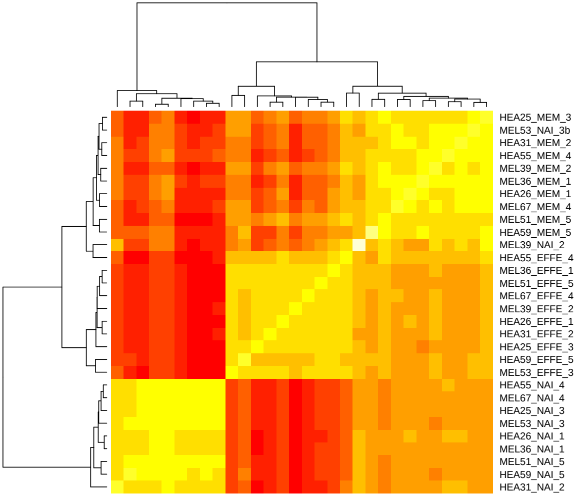
Heatmap of column correlations

```r
corM=cor(as.matrix(Msig3transp))
heatmap(corM,distfun=function(c)as.dist(1-c),labCol = "",labRow="")
```



Heatmap of row correlations

```r
corT=cor(as.matrix(t(Msig3transp)))
heatmap(corT,distfun=function(c)as.dist(1-c),labCol ="")
```



No need to invent a new clustering algorithm

Too many already

Diagnostic methods for numbers of clusters

- Gap statistic
- Elbows in the WSS plots
- Silhouette plot
- Calinski-Harabasz
- Look at the long edges in hierarchical clustering

Summary of clustering methods

- Choices of distances (garbage in, garbage out)
- Choices of assembly of classes criteria
- Choice of number of clusters
- A clustering algorithm will always try to cluster the data, even if the data aren't grouped.

Further reading

1. Tibshirani, Robert, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63.2 (2001): 411-423.