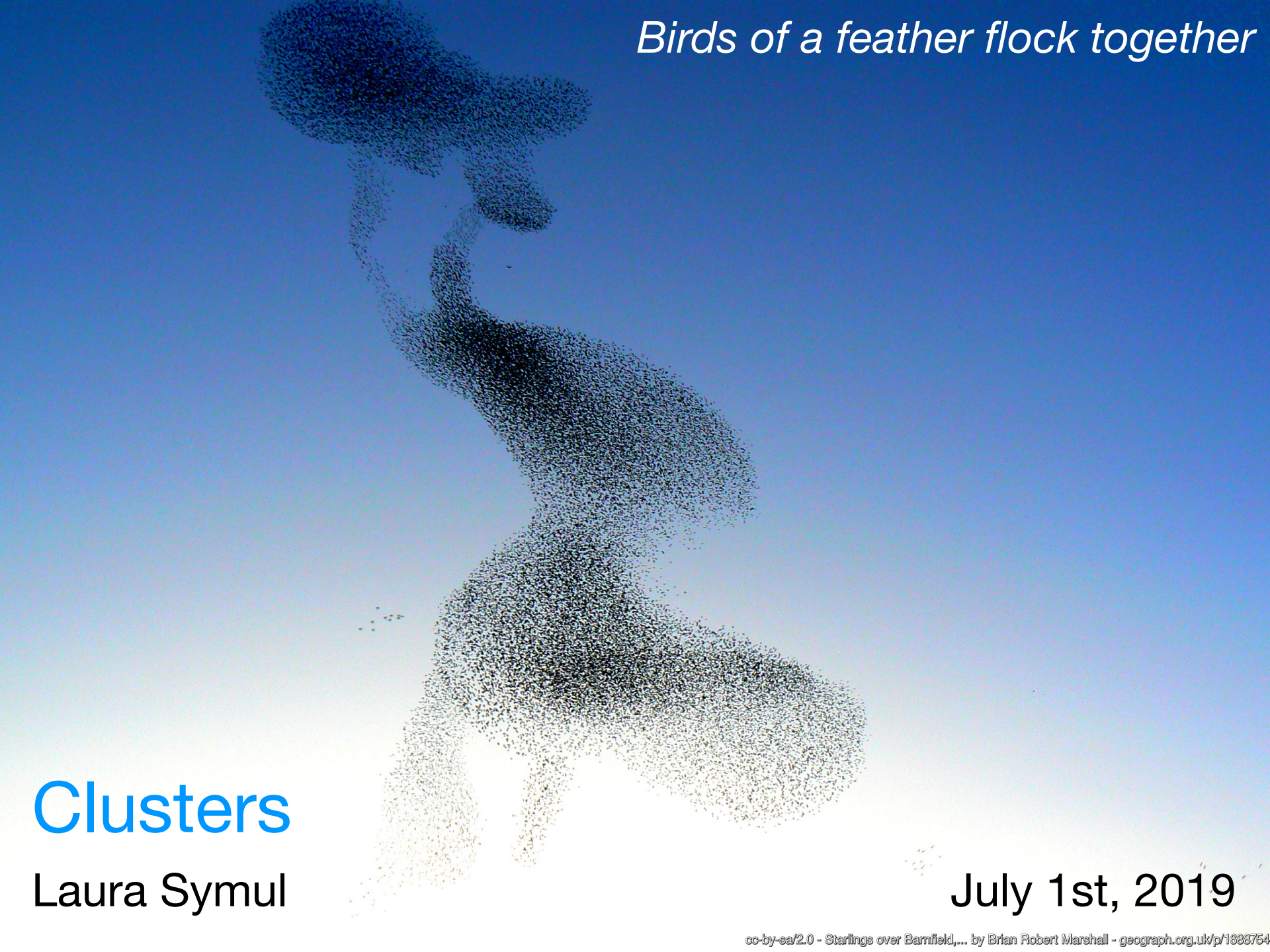


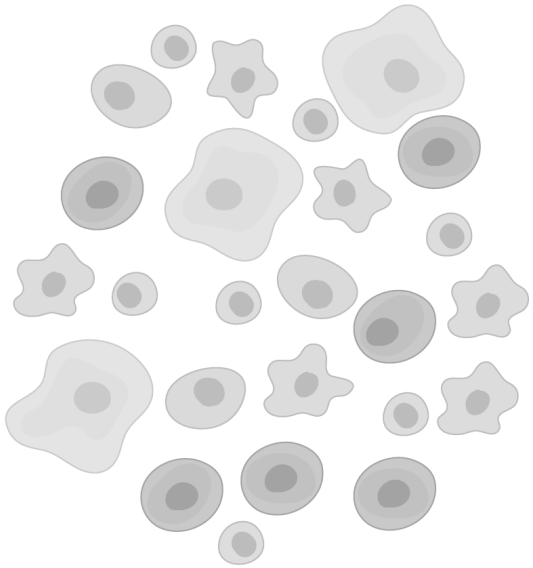
Clusters

Laura Symul

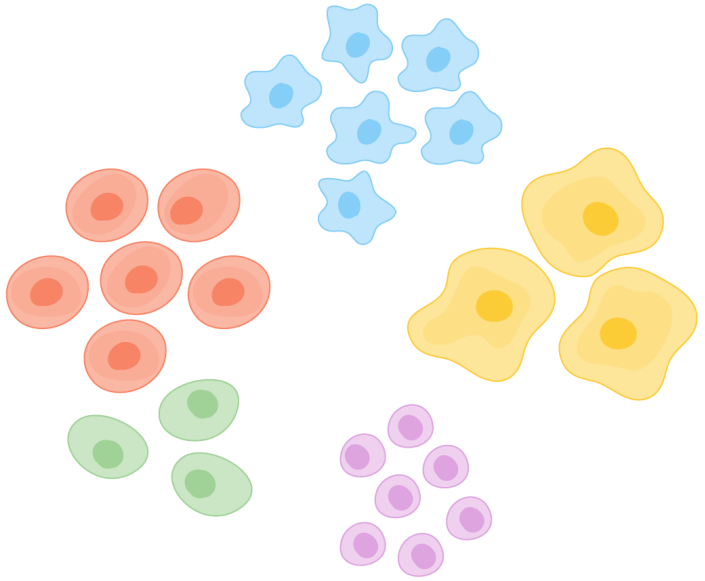
July 1st, 2019

Clustering =
Finding a latent or hidden variable which is not necessarily provided/accessible

Clusters =
Groups in the population (of individuals, cells, gene expression profiles) in which the **individual elements are similar to each others** and **different from the elements of other groups**.



Some **measurements**
(imaging, single cell RNA-seq, etc.)

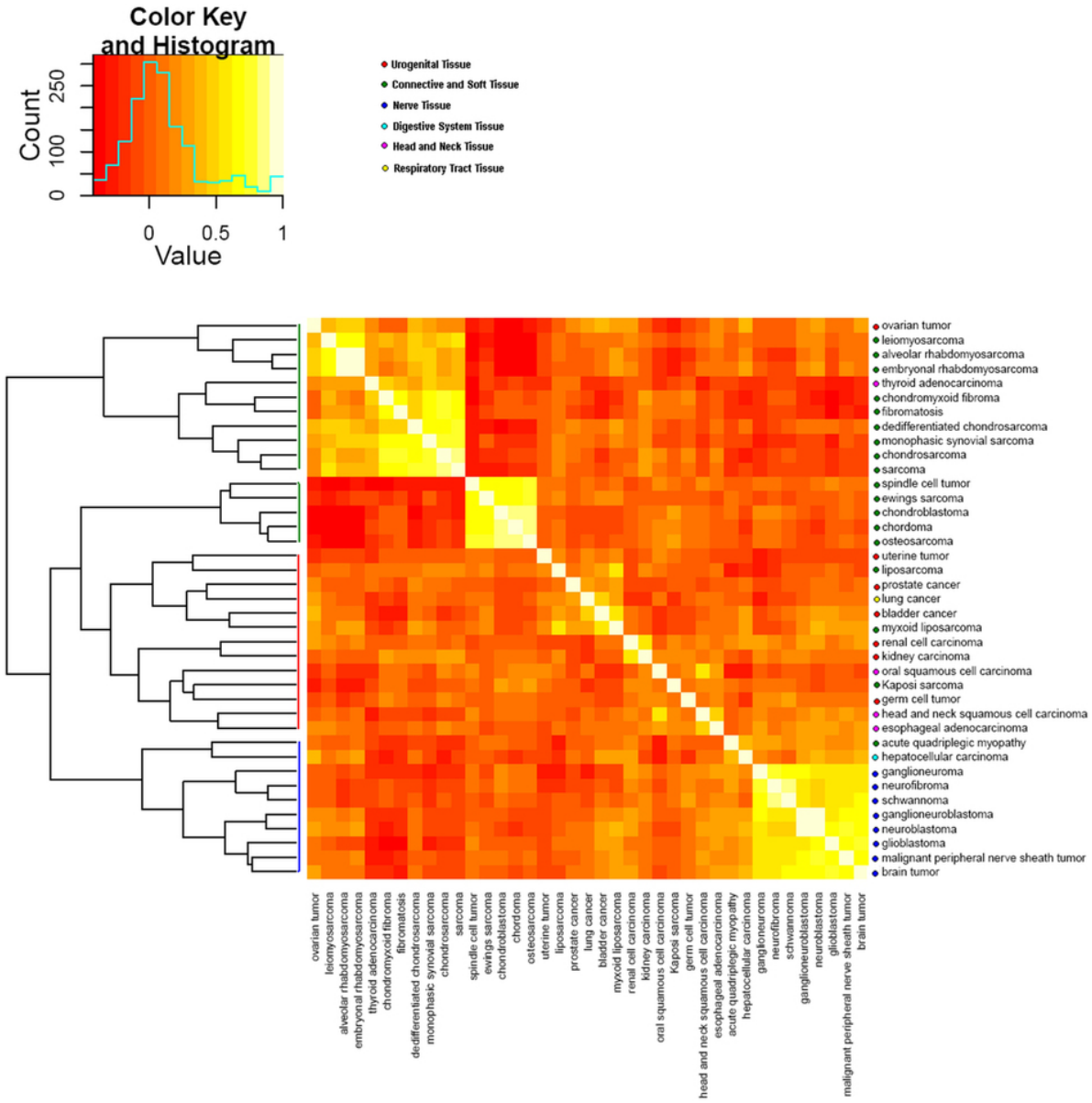


Groups = cell types

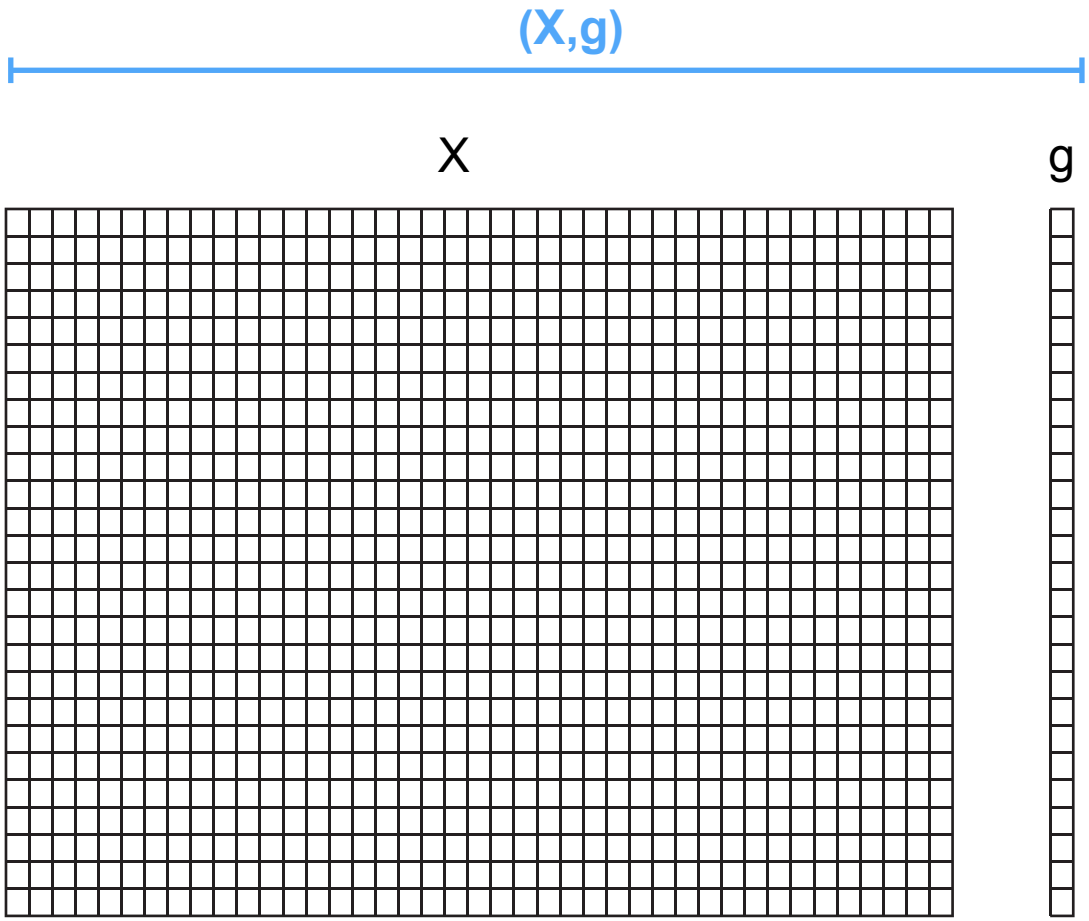
What is the central statistical concept behind any clustering?

Metric of (dis)similarity : a distance

Clusters frequently occur in biology.
Cell type, tissue, tumor type, ethnicities, etc.



Data Augmentation is the aim of clustering



data augmentation methods are methods that add useful but unknown components to the data

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

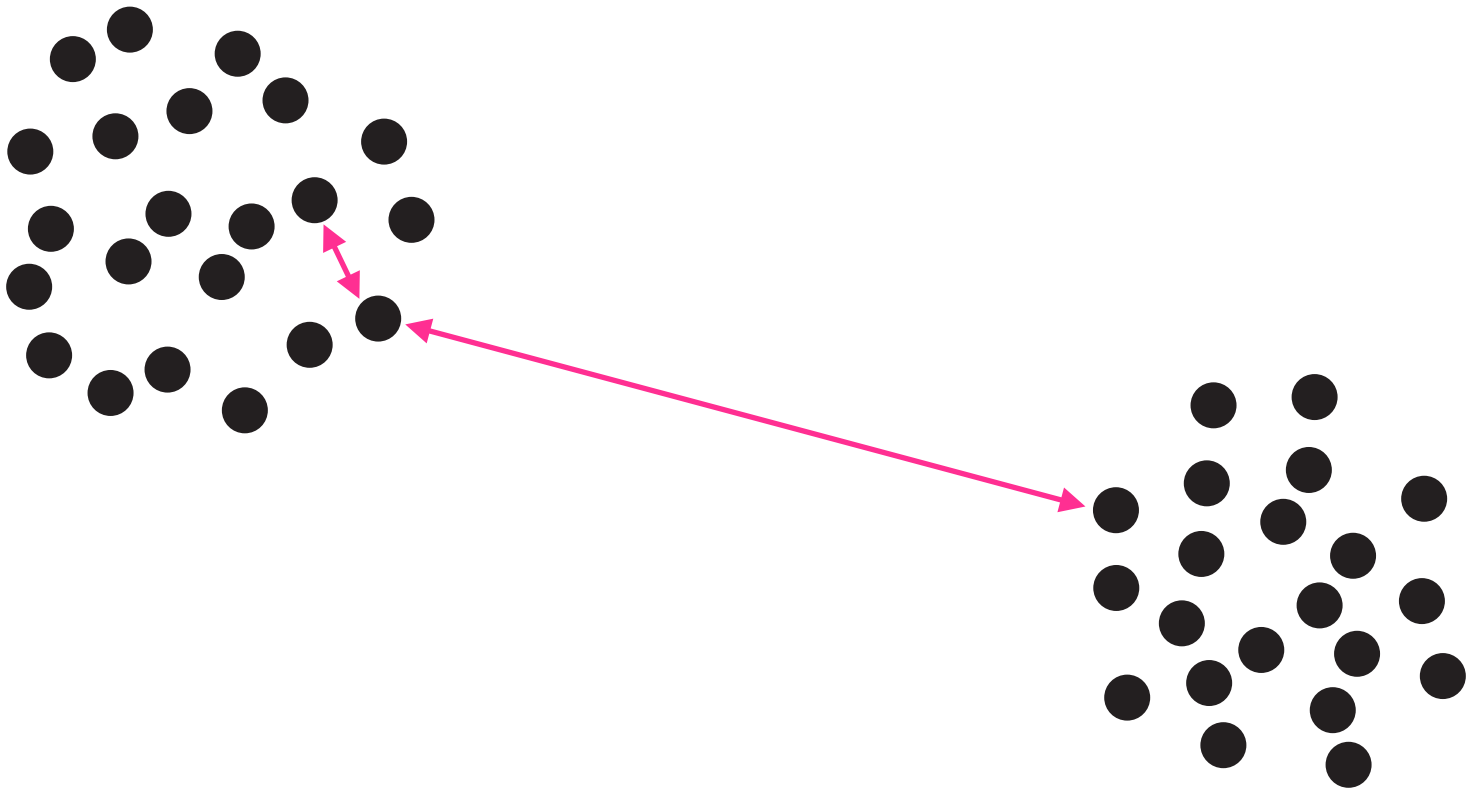
3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

How (dis)similar are 2 elements?

(dis)similarity measure = **distance**

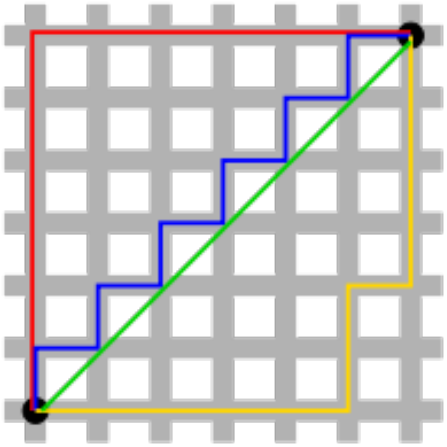
How (dis)similar are 2 elements?



How (dis)similar are 2 elements?

$$
\begin{array}{r}
01000100100 1 \\
00100100100 1 \\
\hline
00100100100 1 \\
110110110110
\end{array}
$$

The choice or definition of **distance depends on the data**



Euclidean distance (L2) $d(A,B) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \ldots + (a_p-b_p)^2}.$

Manhattan distance (L1) $d(A,B) = |a_1-b_1| + |a_2-b_2| + \ldots + |a_p-b_p|.$

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

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

Edit (Hamming) distance

Binary distance

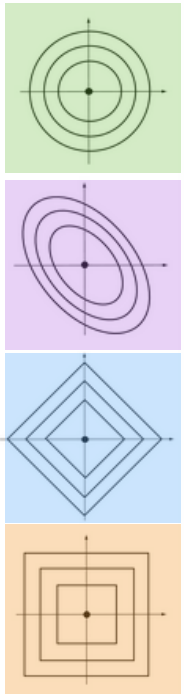
Jaccard distance $d_J(S,T) = 1 - J(S,T) = \frac{f_{01}+f_{10}}{f_{01}+f_{10}+f_{11}}.$

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

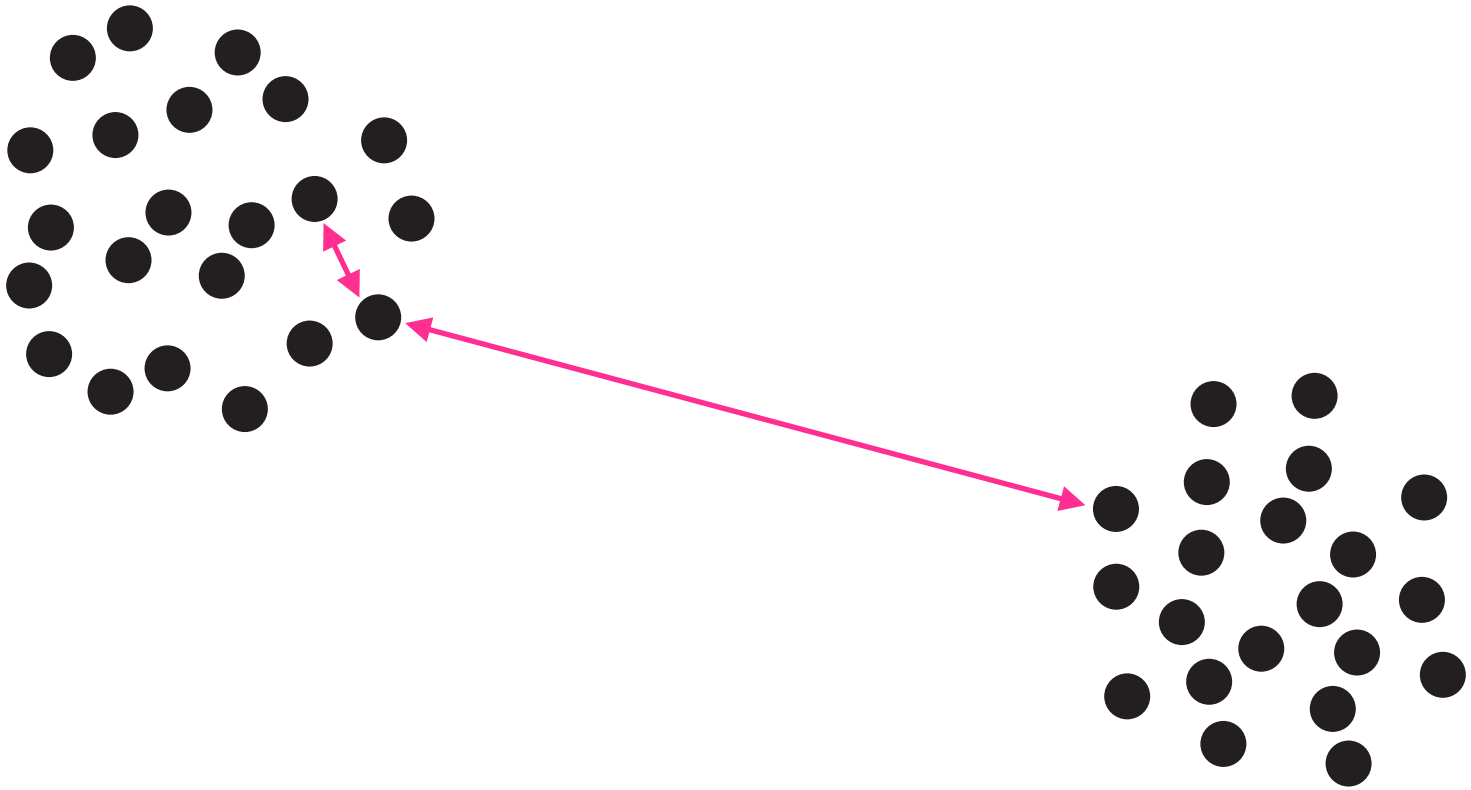
Weighted Euclidean distance

Mahalanobis distance

…



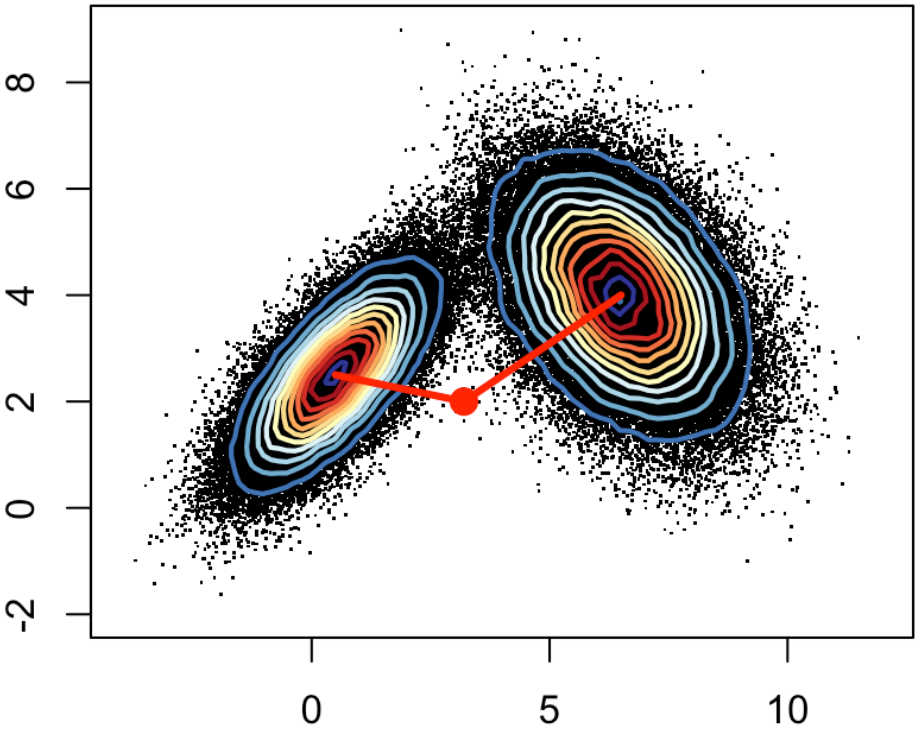
Euclidean distance



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

Weighted Euclidean distance

Is the **red point**, a “new” datapoint, closer to the cluster on the right or the cluster on the left?



The weight on the dimensions depends on the cluster

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

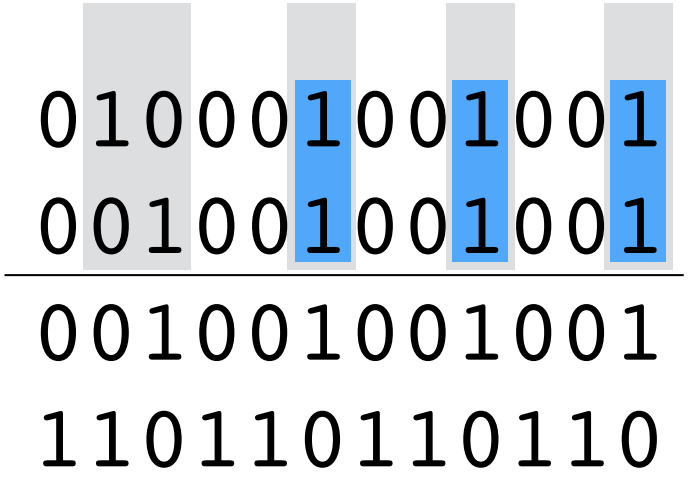
4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Binary distance

$$
\begin{array}{r}
0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\
0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\
\hline
0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1 \\
1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0
\end{array}
$$

$$1- \frac{\text{The sum of elements that are the same}}{\text{The number of elements}}$$

Jaccard distance



The **co-occurrence** is more important than the co-absence

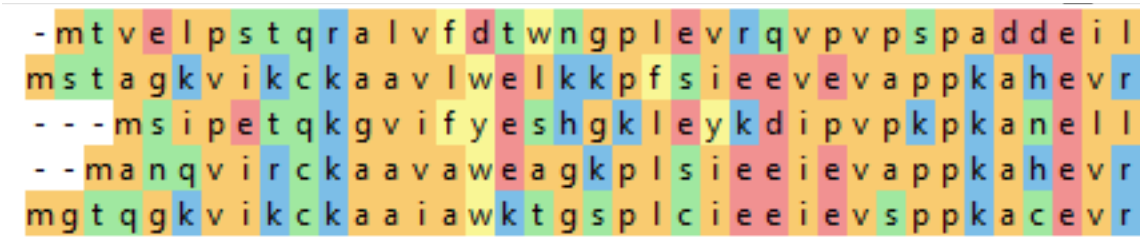
$$1 - \frac{\text{The sum of 1's that overlap}}{\text{The union of 1's}}$$

Initially developed for ecological data for the occurrence of traits or features

Edit (Hamming) distance

The number of edits that needs to be done.

This could be applied to nucleotide or amino acid sequences…



…although in that case, the **different character substitutions** are usually associated with **different contributions to the distance** (to account for physical or evolutionary similarity), and **deletions and insertions** may also be allowed

Distances in R

```
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)
```

Euclidean

```
dist(mat)
```

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

Binary

```
dist(mat, method = "binary")
```

```
##           mx        my
## my 0.6000000
## mz 0.6666667 0.5000000
```

```
mut = read.csv("../data/HIVmutations.csv")
mut[1:3, 10:16]
```

```
##   p32I p33F p34Q p35G p43T p46I p46L
## 1    0    1    0    0    0    0    0
## 2    0    1    0    0    0    1    0
## 3    0    1    0    0    0    0    0
```

```
library("vegan")
mutJ = vegdist(mut, "jaccard")
mutC = sqrt(2 * (1 - cor(t(mut))))
```

Jaccard

```
mutJ
```

```
##       1     2     3     4
## 2 0.800
## 3 0.750 0.889
## 4 0.900 0.778 0.846
## 5 1.000 0.800 0.889 0.900
```

Correlation-based

```
as.dist(mutC)
```

```
##      1    2    3    4
## 2 1.19
## 3 1.10 1.30
## 4 1.32 1.13 1.30
## 5 1.45 1.19 1.30 1.32
```

Distances in R
phyloseq package

phyloseq: analyze microbiome census data using R

The phyloseq package is a tool to import, store, analyze, and graphically display complex phylogenetic sequencing data that has already been clustered into Operational Taxonomic Units (OTUs), especially when there is associated sample data, phylogenetic tree, and/or taxonomic assignment of the OTUs.

```
> library("phyloseq")
> distanceMethodList
$UniFrac
[1] "unifrac"  "wunifrac"

$DPCoA
[1] "dpcoa"

$JSD
[1] "jsd"

$vegdist
 [1] "manhattan"  "euclidean"  "canberra"   "bray"       "kulczynski" "jaccard"    "gower"      "altGower"   "morisita"
[10] "horn"       "mountford"  "raup"       "binomial"   "chao"       "cao"

$betadiver
 [1] "w"   "-1"  "c"   "wb"  "r"   "I"   "e"   "t"   "me"  "j"   "sor" "m"   "-2"  "co"  "cc"  "g"   "-3"  "l"   "19"  "hk"
[21] "rlb" "sim" "gl"  "z"

$dist
[1] "maximum"   "binary"    "minkowski"

$designdist
[1] "ANY"

>
```

Almost 50 different distances

http://joey711.github.io/phyloseq/distance.html

If you **really** need to define a **new distance**, it must…

Definition [edit]

A **metric space** is an ordered pair (M, d) where M is a set and d is a metric on M, i.e., a function

$$d\colon M \times M \to \mathbb{R}$$

such that for any $x, y, z \in M$, the following holds:[2]

1. $d(x, y) = 0 \Leftrightarrow x = y$ identity of indiscernibles
2. $d(x, y) = d(y, x)$ symmetry
3. $d(x, z) \leq d(x, y) + d(y, z)$ subadditivity or triangle inequality

Given the above three axioms, we also have that $d(x, y) \geq 0$ for any $x, y \in M$. This is deduced as follows:

https://en.wikipedia.org/wiki/Metric_space

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

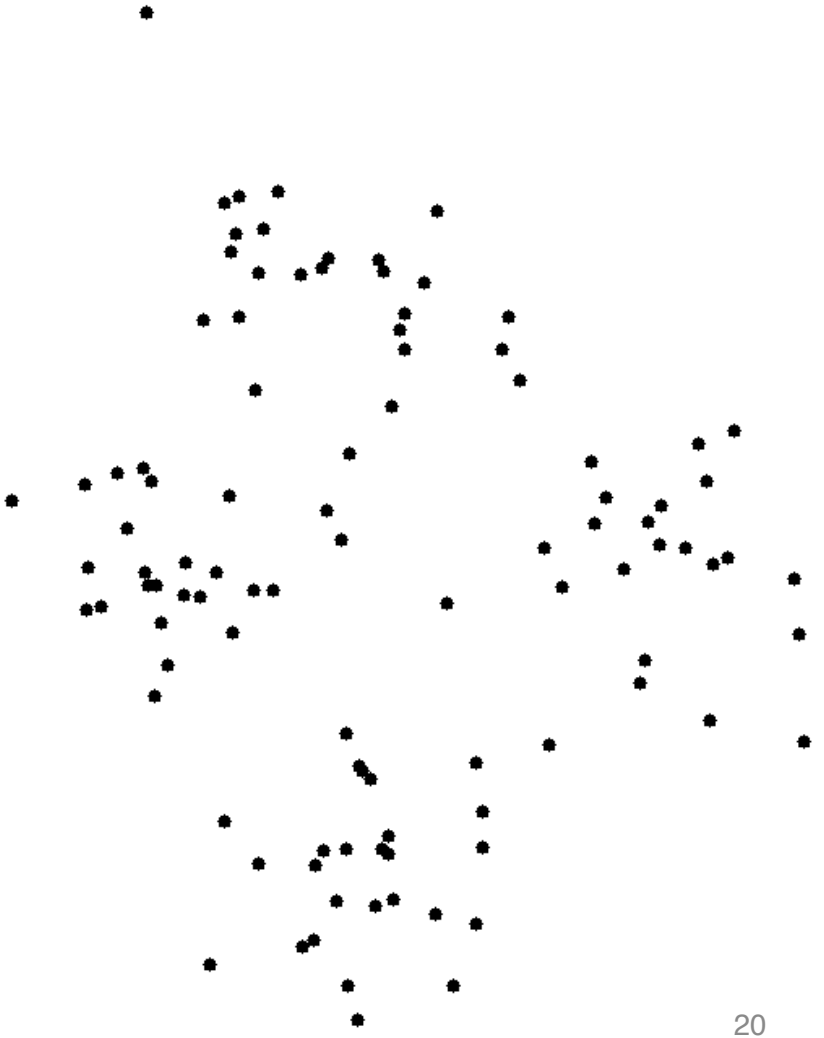
Non-parametric clustering methods (k-methods, top-down)

k-medoids

PAM (Partitioning **A**round (k) **M**edoids**)**

k-means

1. Starts from a matrix of p features measured on a set of n observations.
2. **Randomly pick k distinct cluster centers** out of the n observations ("seeds").
3. **Assign each of the** remaining **observations** to the group to whose center it is the closest.
4. **For each group, choose a new center** from the observations in the group, such that the sum of the distances of group members to the center is minimal; this is called the ***medoid***.
5. **Repeat** Steps 3 and 4 **until the groups stabilize**.



http://shabal.in/visuals/kmeans/4.html

Non-parametric clustering methods (k-methods, top-down)

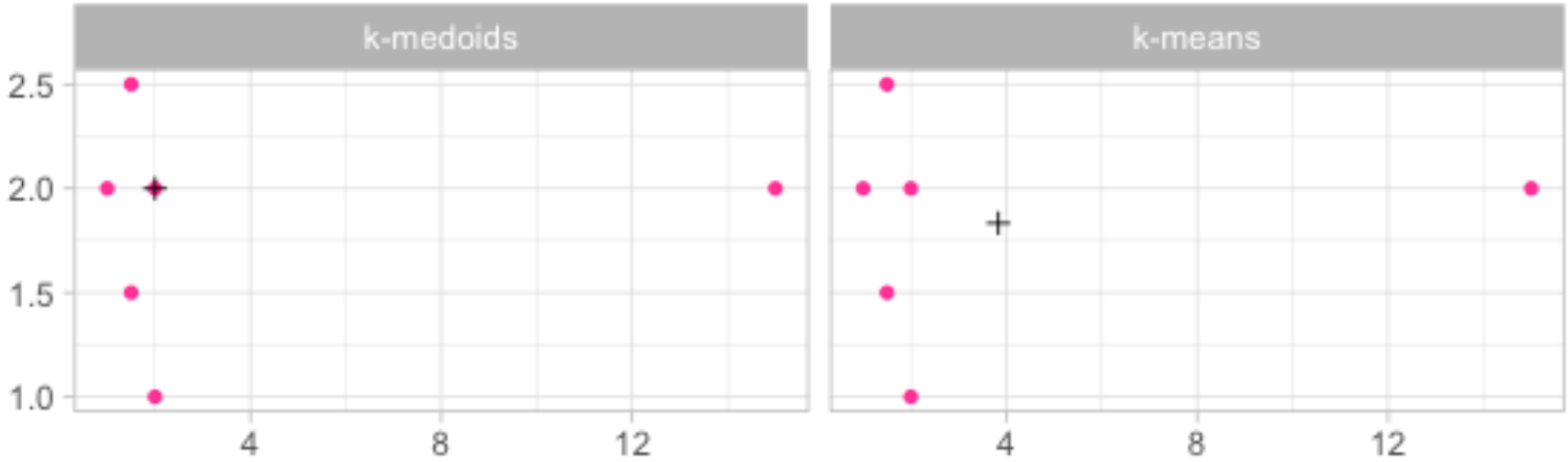
The difference between k-medoids and k-means is at step 4, when choosing the new centers

k-medoids

The new centers are selected **among the observations**

k-means

The new centers are **computed** as the **arithmetic mean** between the observations of each group



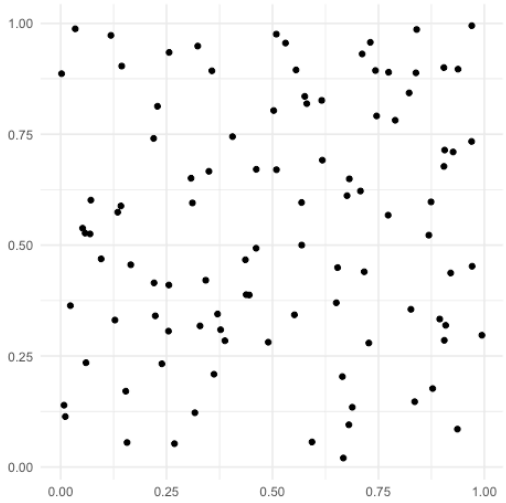
Non-parametric clustering methods (k-methods, top-down)

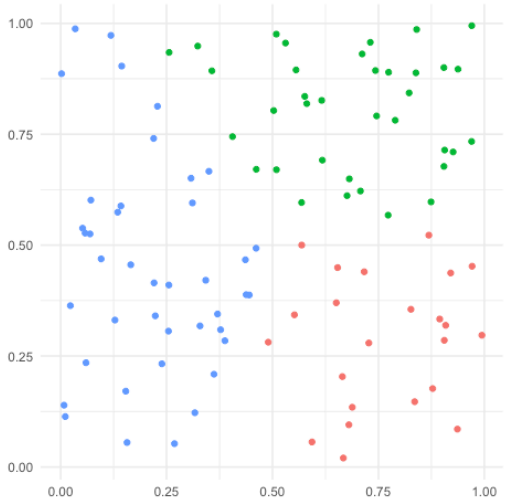
```
Xmat=matrix(runif(200),ncol=2)
nk=3
cents=matrix(runif(2*nk),ncol=2)

out=kmeans(Xmat,centers = nk) # give the # of clusters
out=kmeans(Xmat,centers = cents) # give the seeds (initial center of clusters)

X = data.frame(x = Xmat[,1], y = Xmat[,2], cluster = factor(out$cluster))

g = ggplot(X, aes(x = x, y = y)) + xlab("") + ylab("") + guides(col = FALSE)
g + geom_point()
g + geom_point(aes(col = cluster))
```






```
out1=kmeans(Xmat,cents,iter.max=1)
out2=kmeans(Xmat,cents,iter.max=2)
out3=kmeans(Xmat,cents,iter.max=3)
```

Non-parametric clustering methods (k-methods, top-down)

1. Starts from a matrix of p features measured on a set of n observations.
2. **Randomly pick** k **distinct cluster centers** out of the n observations ("seeds").
3. **Assign each of the** remaining **observations** to the group to whose center it is the closest.
4. **For each group, choose a new center** from the observations in the group, such that the sum of the distances of group members to the center is minimal; this is called the ***medoid***.
5. **Repeat** Steps 3 and 4 **until the groups stabilize**.

Is the method robust?
i.e. does it give always the same result?

How can we improve that?

http://shabal.in/visuals/kmeans/4.html

Strong Forms & Tight Clusters

Dynamical Clusters

Repeats the process many times with **different seeds** (initial centers) to build `**strong forms**' which are groups of observations that end up in the same classes for most possible initial configurations.

Diday and Brito, 1989

Resampling

Repeats the process many times on subsamples of the dataset to create `**tight clusters**' which are groups of observations that are almost always grouped together.

Tseng and Wong, 2015

In R, see package ***clusterExperiment***

Goals for this lecture

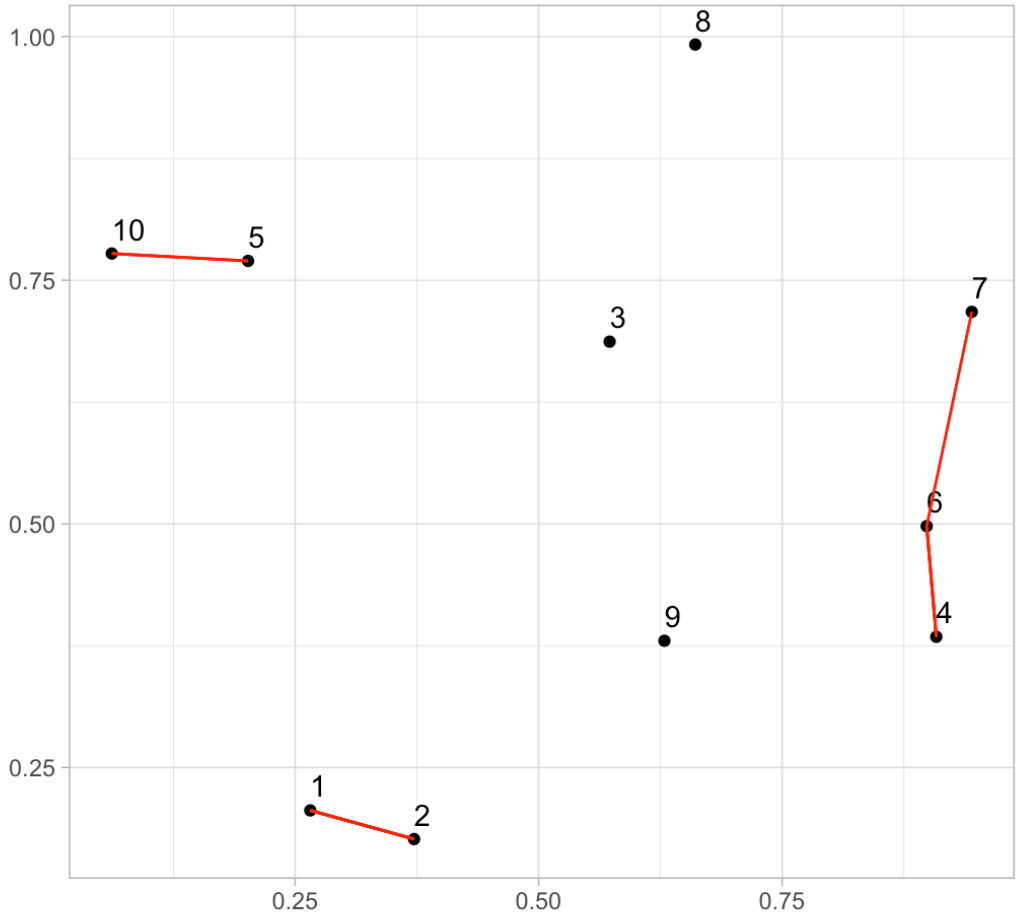
1. Review measures of **(dis)similarity** that help us define clusters.

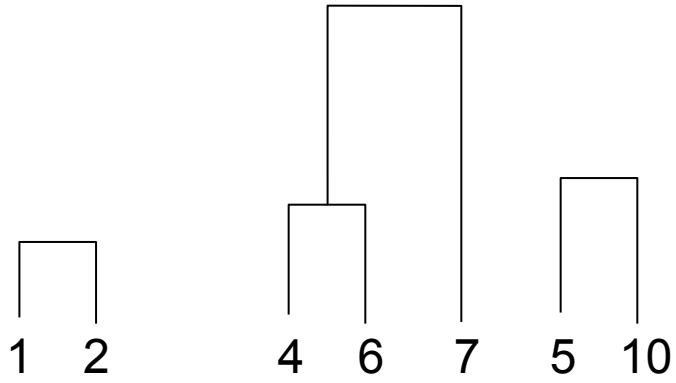
2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

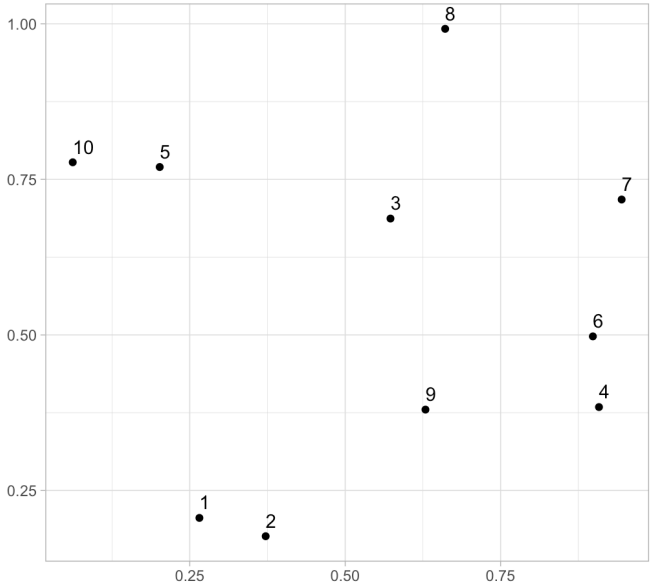
4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Hierarchical Clustering
(bottom-up)





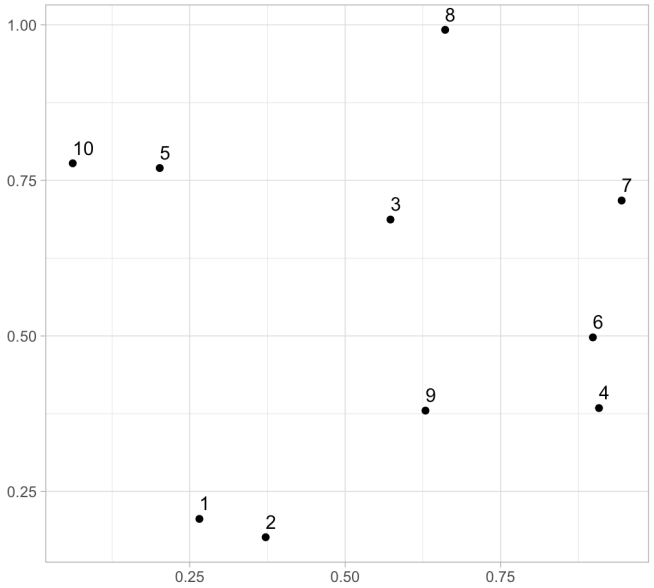
Hierarchical Clustering (bottom-up)




```
set.seed(1)
Xmat=matrix(runif(20),ncol=2)
d = dist(Xmat)
```

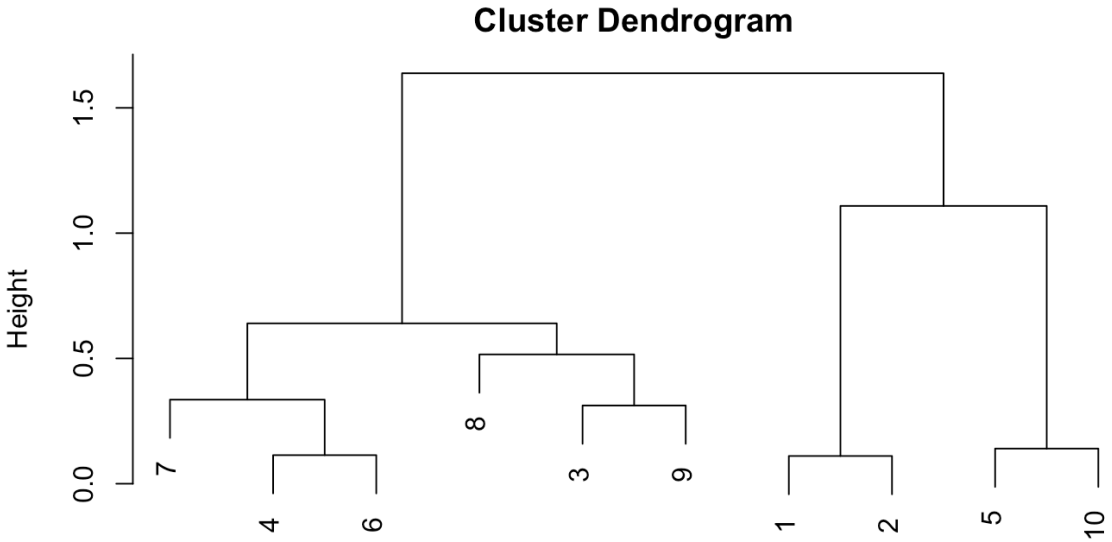
```
> round(d,3)
       1     2     3     4     5     6     7     8     9
2  0.111
3  0.571 0.549
4  0.667 0.575 0.452
5  0.567 0.617 0.380 0.805
6  0.697 0.617 0.377 0.114 0.748
7  0.850 0.788 0.373 0.336 0.745 0.225
8  0.880 0.865 0.317 0.656 0.510 0.548 0.395
9  0.403 0.328 0.312 0.279 0.578 0.294 0.462 0.613
10 0.607 0.676 0.519 0.933 0.140 0.882 0.885 0.636 0.693
```

Hierarchical Clustering
(bottom-up)




```
set.seed(1)
Xmat=matrix(runif(20),ncol=2)
d = dist(Xmat)
```

```
clust = hclust(d, method = "ward.D")
plot(clust)
```



Goals for this lecture

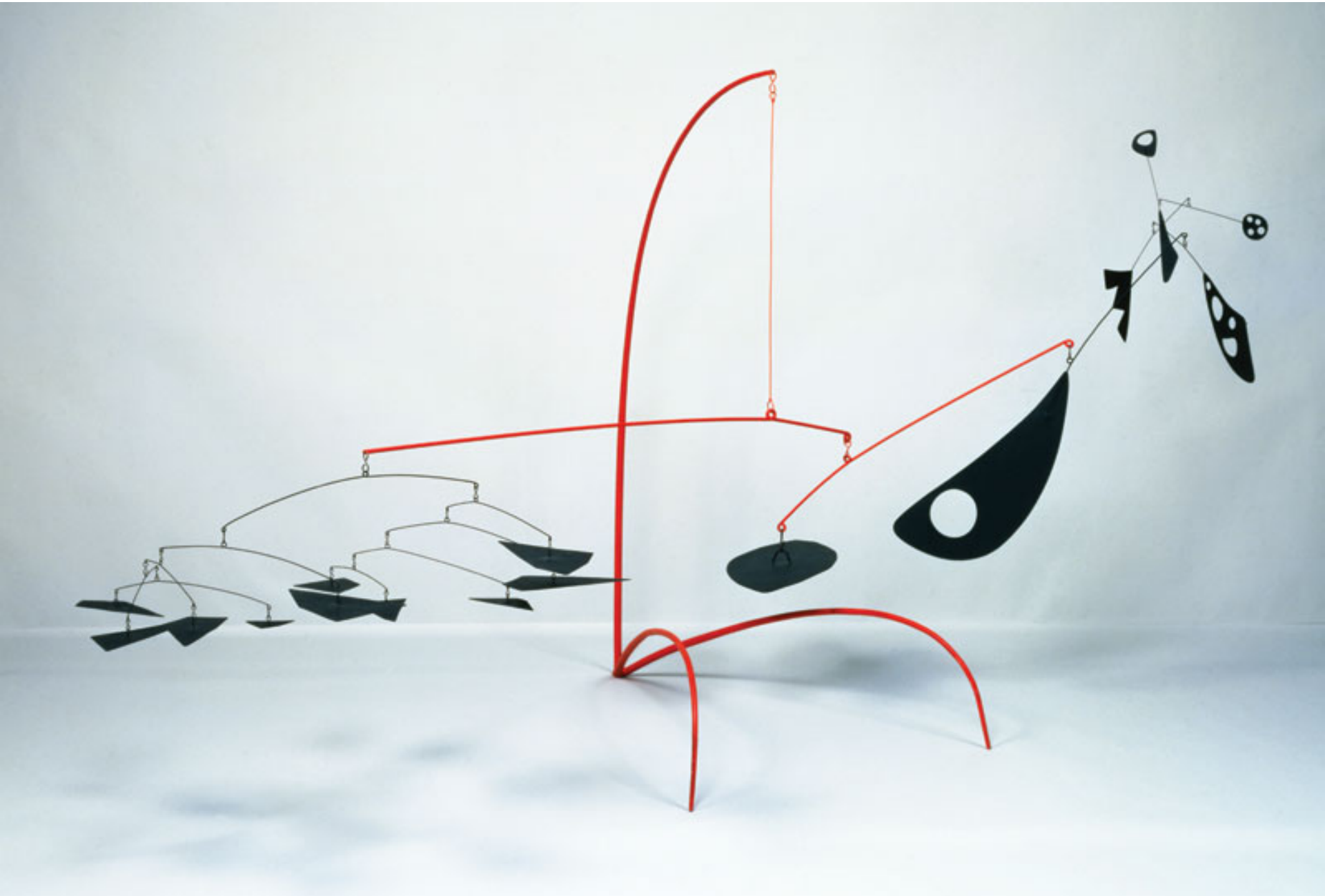
1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

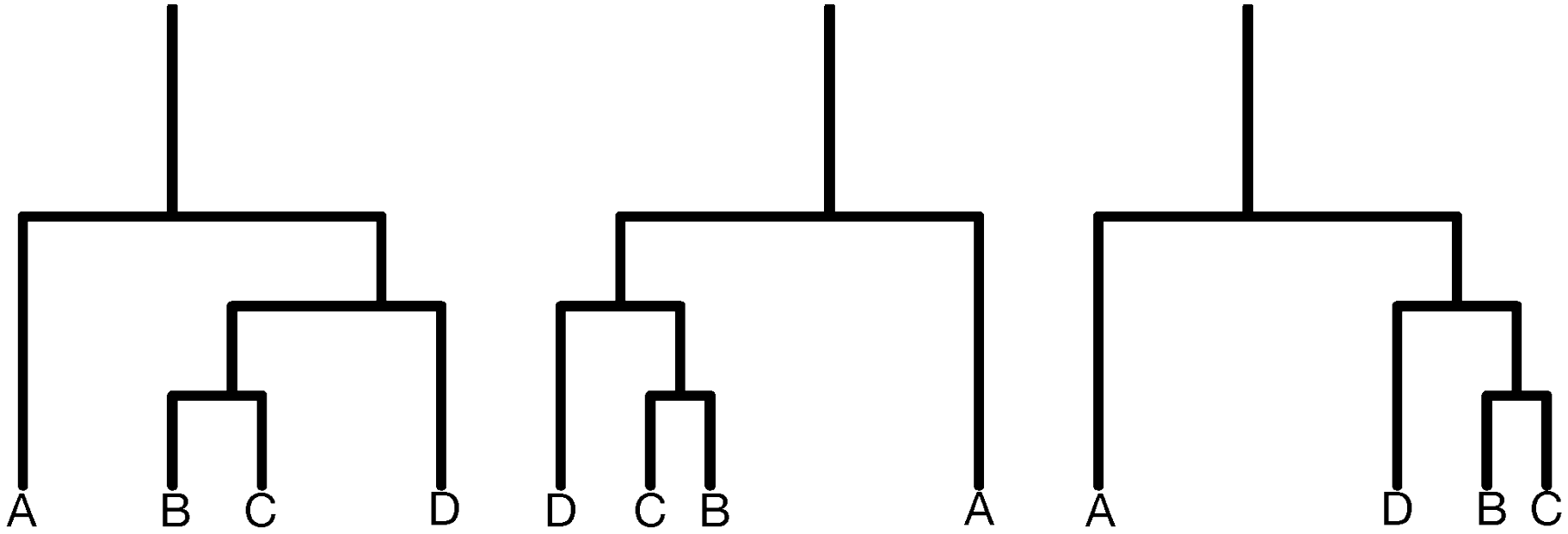
3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

The ordering of the branches is not unique



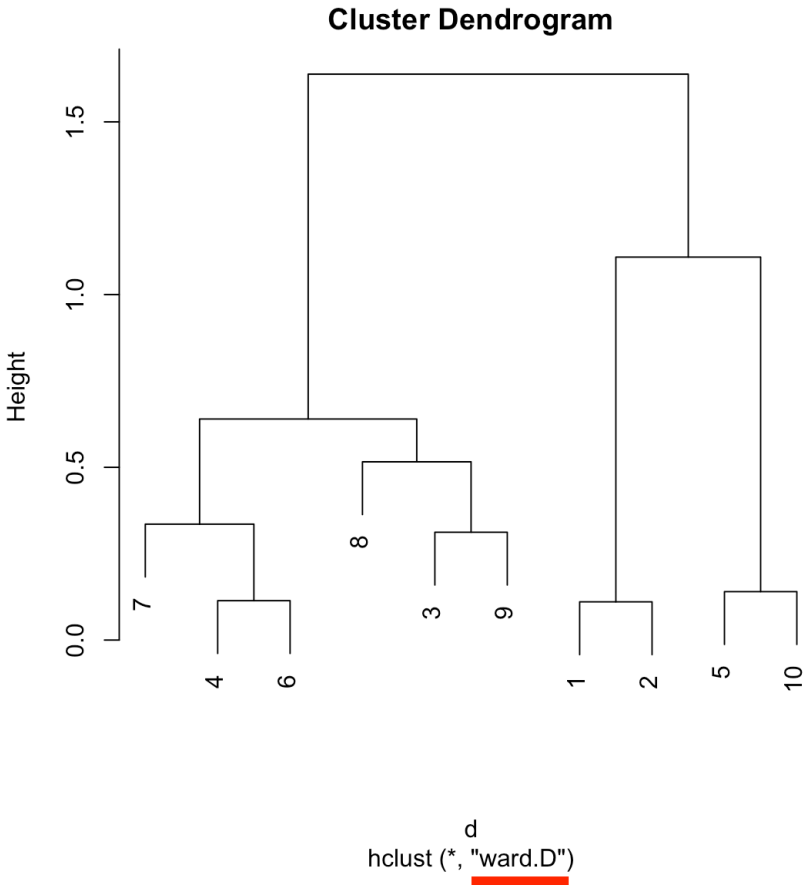
The ordering of the branches is not unique

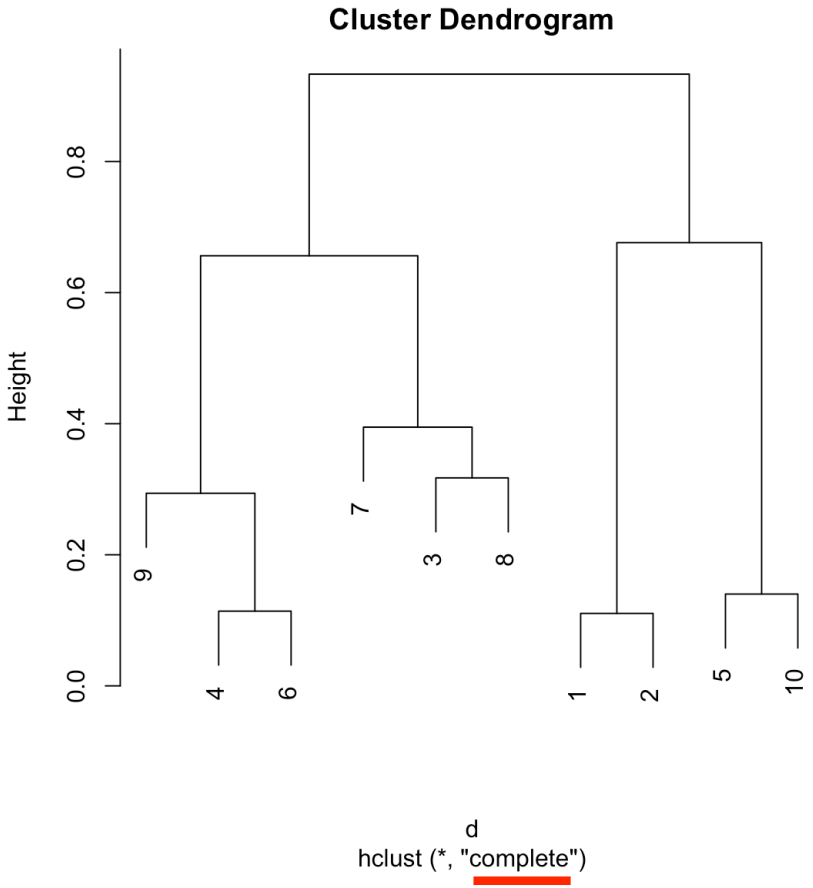


What matters is the "**order of the agglomerations**" and the **length between leaves**

Different options to agglomerate

```
clust = hclust(d, method = "ward.D")
plot(clust)
```

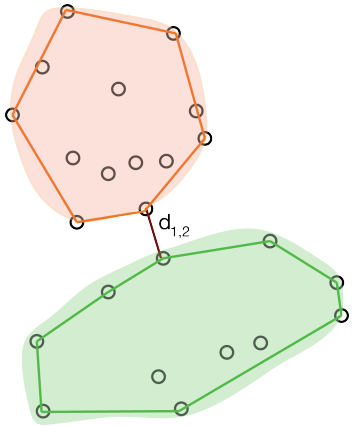




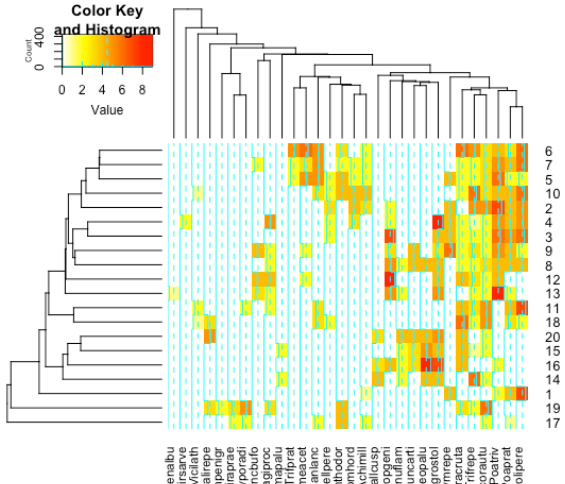
Can you think of different ways to agglomerate a new point (or subgroup) ?

Different options to agglomerate

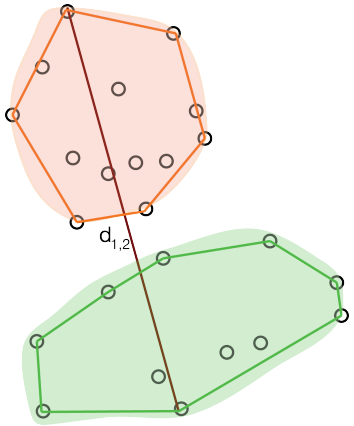
Single linkage



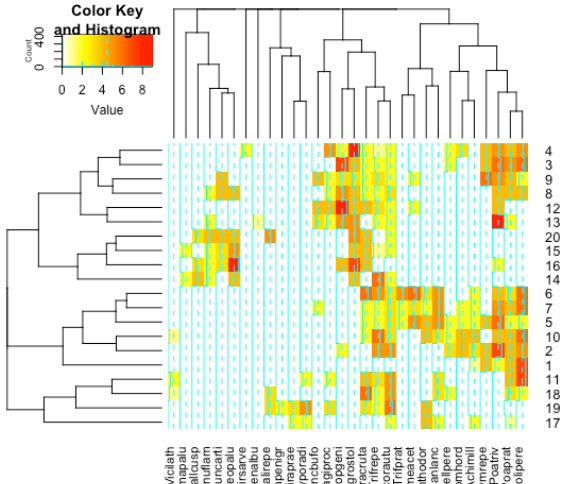
Good for recognizing the number of clusters
But “combs”



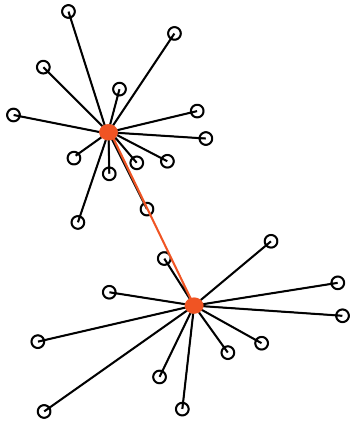
Maximal linkage



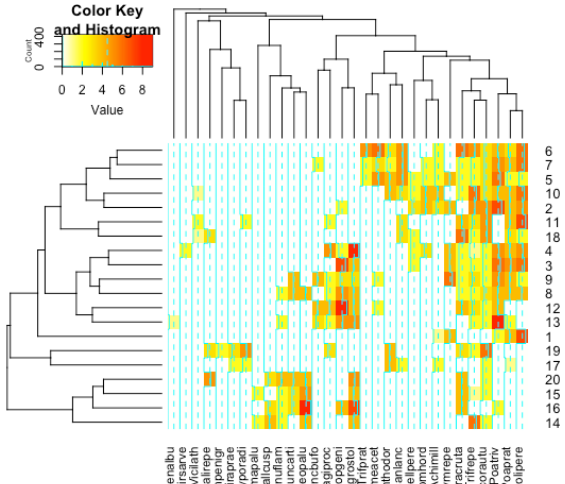
Compact classes
But one observation can alter groups



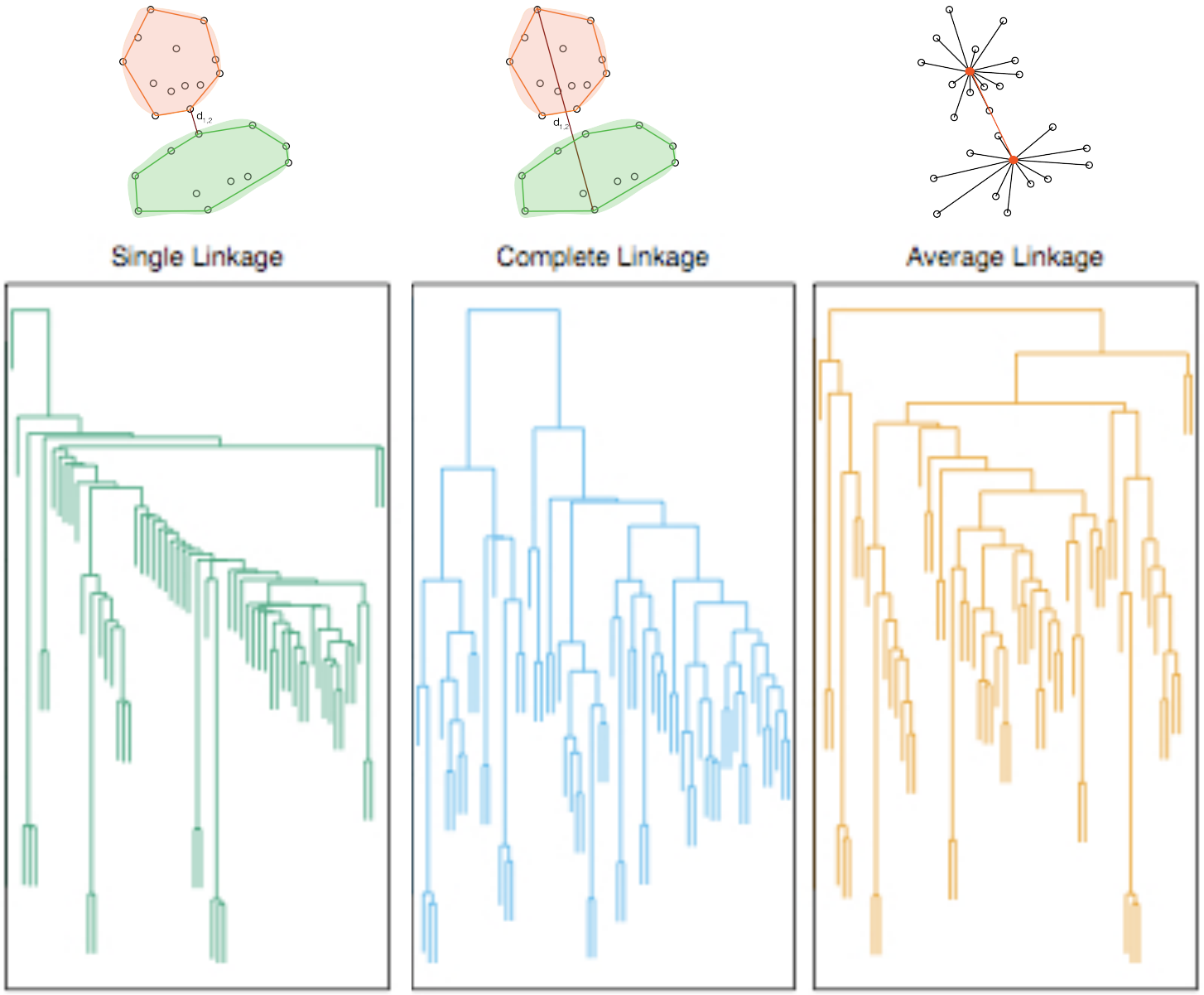
Medoids



More robust to outliers



Different options lead to **different tree shapes**

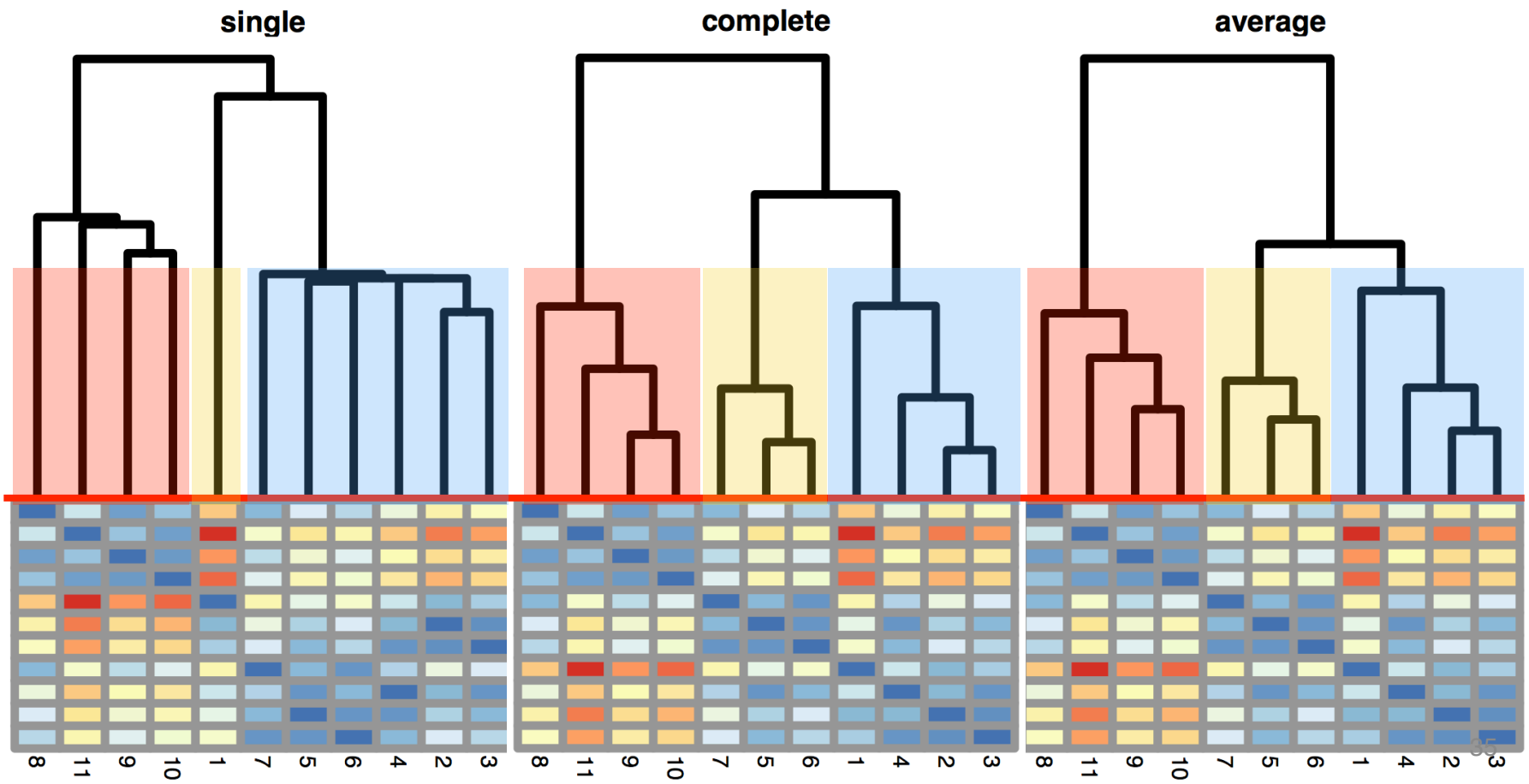


of clusters <-> max dist between groups

You cut the tree such that

- you have a given number of cluster
- at a given height (distance between the groups)

How do you choose the number of clusters?



Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Clustering methods WILL find clusters

… even if there are none

How do we validate the clusters?

How do we find the optimal number of cluster?

We want to
Minimize the distance between the points of a cluster (cohesion),
Maximizing the distance between clusters (separation).

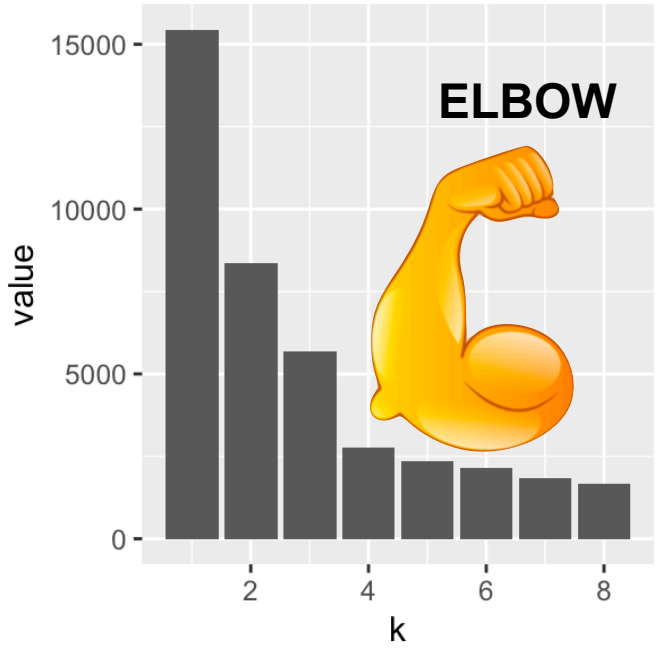
There are different ways to assess this.

- > WSS (Within-group sum of squares)
- Elbow method
- Calinski-Harabasz index
- Gap statistic

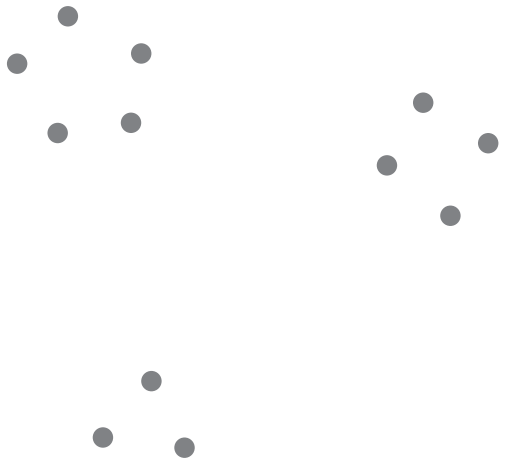
- Silhouette method

WSS

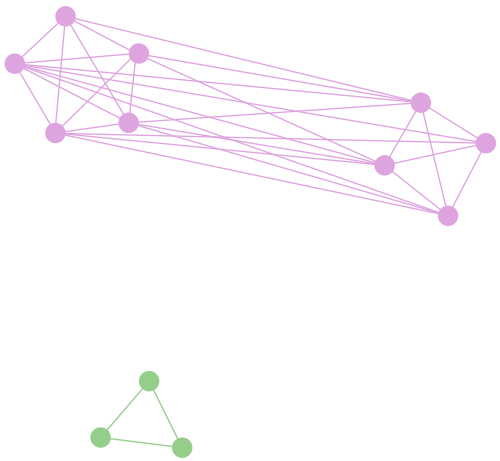
$$\mathrm{WSS}_k = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i, \bar{x}_\ell)$$



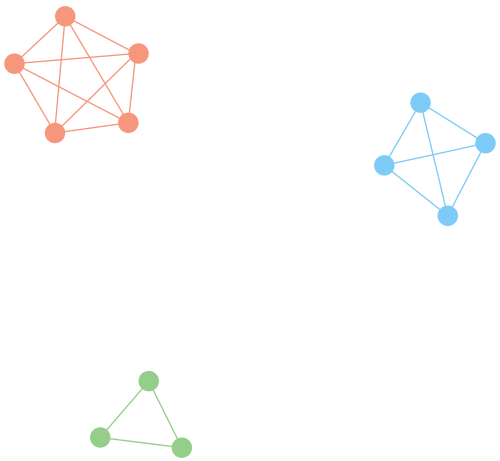
Data



2 clusters



3 clusters



Calinski-Harabasz index

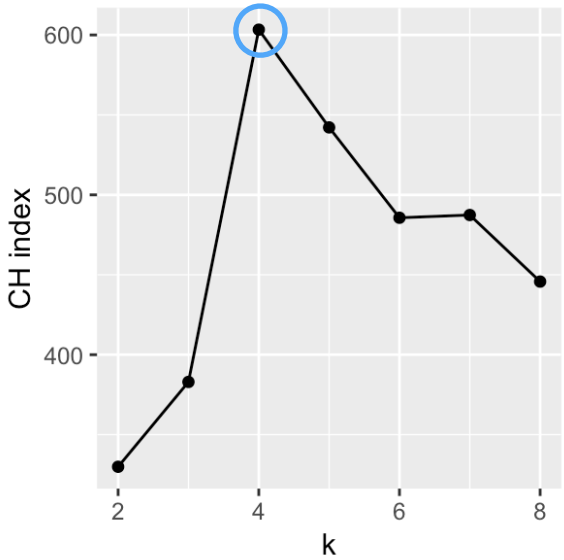
We want to maximize

$$\mathrm{CH}(k) = \frac{\text{Between-groups sum of squares}}{\text{Within-groups sum of squares}}$$

We want to maximize

We want to minimize

$$\mathrm{CH}(k) = \frac{\mathrm{BSS}_k}{\mathrm{WSS}_k} \times \frac{N-k}{N-1} \qquad \text{where} \quad \mathrm{BSS}_k = \sum_{\ell=1}^{k} n_\ell (\bar{x}_\ell - \bar{x})^2,$$



where $\bar{x}$ is the overall center of mass (average point).

The Gap statistic

Algorithm for computing the gap statistic (Tibshirani, Walther, and Hastie 2001):

1. Cluster the data with k clusters and compute WSS_k for the various choices of k.

2. Generate B plausible reference data sets, using Monte Carlo sampling from a homogeneous distribution and redo Step 1 above for these new simulated data. This results in B new within-sum-of-squares for simulated data W^*_{kb}, for $b = 1, \ldots, B$.

3. Compute the $\mathrm{gap}(k)$-statistic:

$$\mathrm{gap}(k) = \bar{l}_k - \log \mathrm{WSS}_k \quad \text{with} \quad \bar{l}_k = \frac{1}{B}\sum_{b=1}^{B} \log W^*_{kb}$$

Note that the first term is expected to be bigger than the second one if the clustering is good (i.e., the WSS is smaller); thus the gap statistic will be mostly positive and we are looking for its highest value.

4. We can use the standard deviation

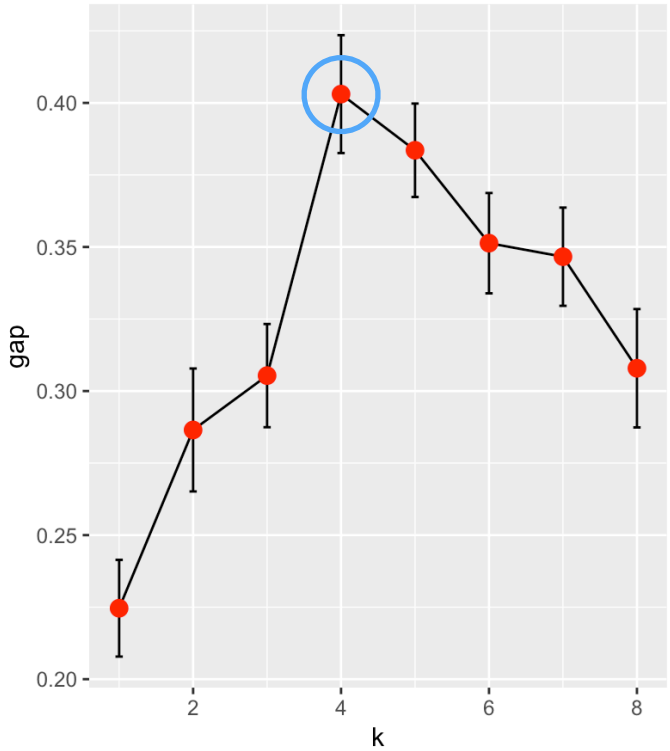
$$\mathrm{sd}_k^2 = \frac{1}{B-1}\sum_{b=1}^{B}\left(\log(W^*_{kb}) - \bar{l}_k\right)^2$$

to help choose the best k. Several choices are available, for instance, to choose the smallest k such that

$$\mathrm{gap}(k) \geq \mathrm{gap}(k+1) - s'_{k+1} \qquad \text{where } s'_{k+1} = \mathrm{sd}_{k+1}\sqrt{1 + 1/B}.$$

The packages **cluster** and **clusterCrit** provide implementations.

Monte Carlo



The Silhouette method

For each datapoint we compare

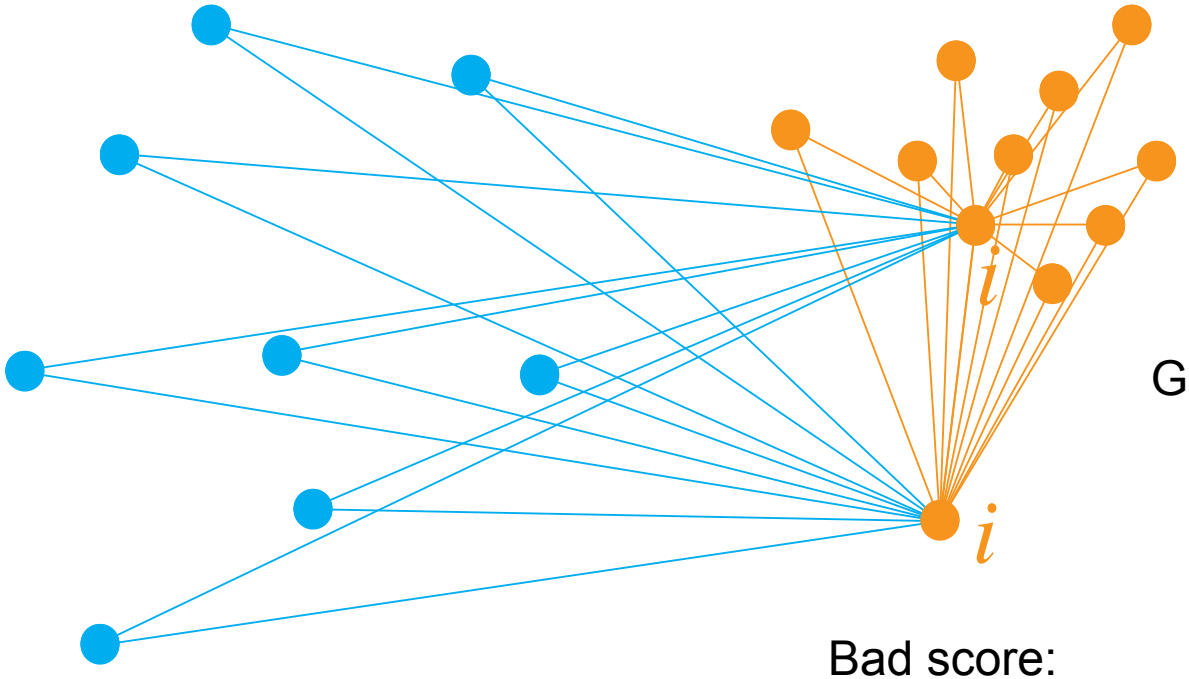
- the **average distance** to all the points of its **own cluster**
- the **average distance** to all the points of the **nearest cluster**

a_i = average of ——

b_i = average of ——

$$s_i = \frac{b_i - a_i}{max\{a_i, b_i\}}$$

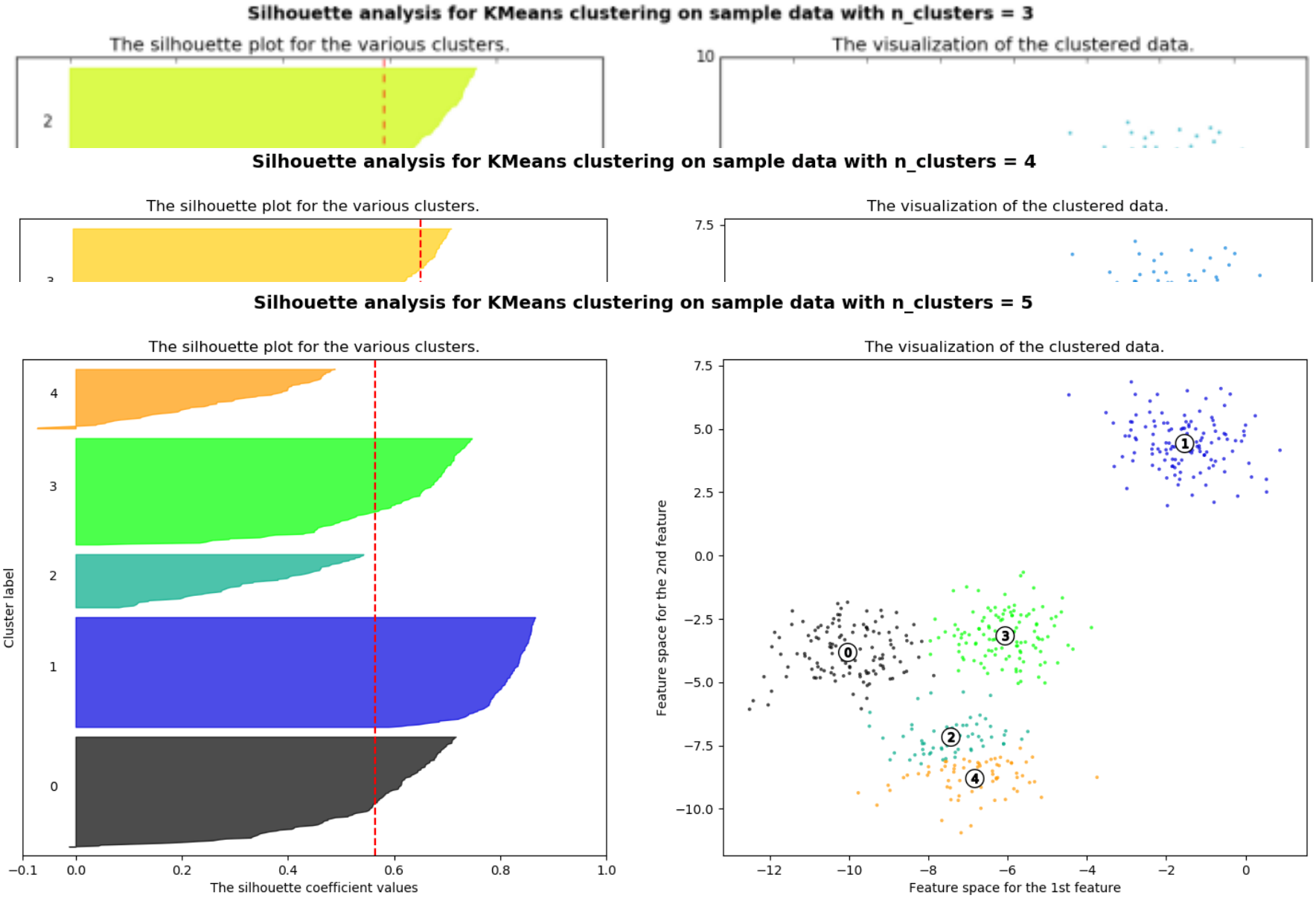
$$-1 \leq s_i \leq 1$$



Good score

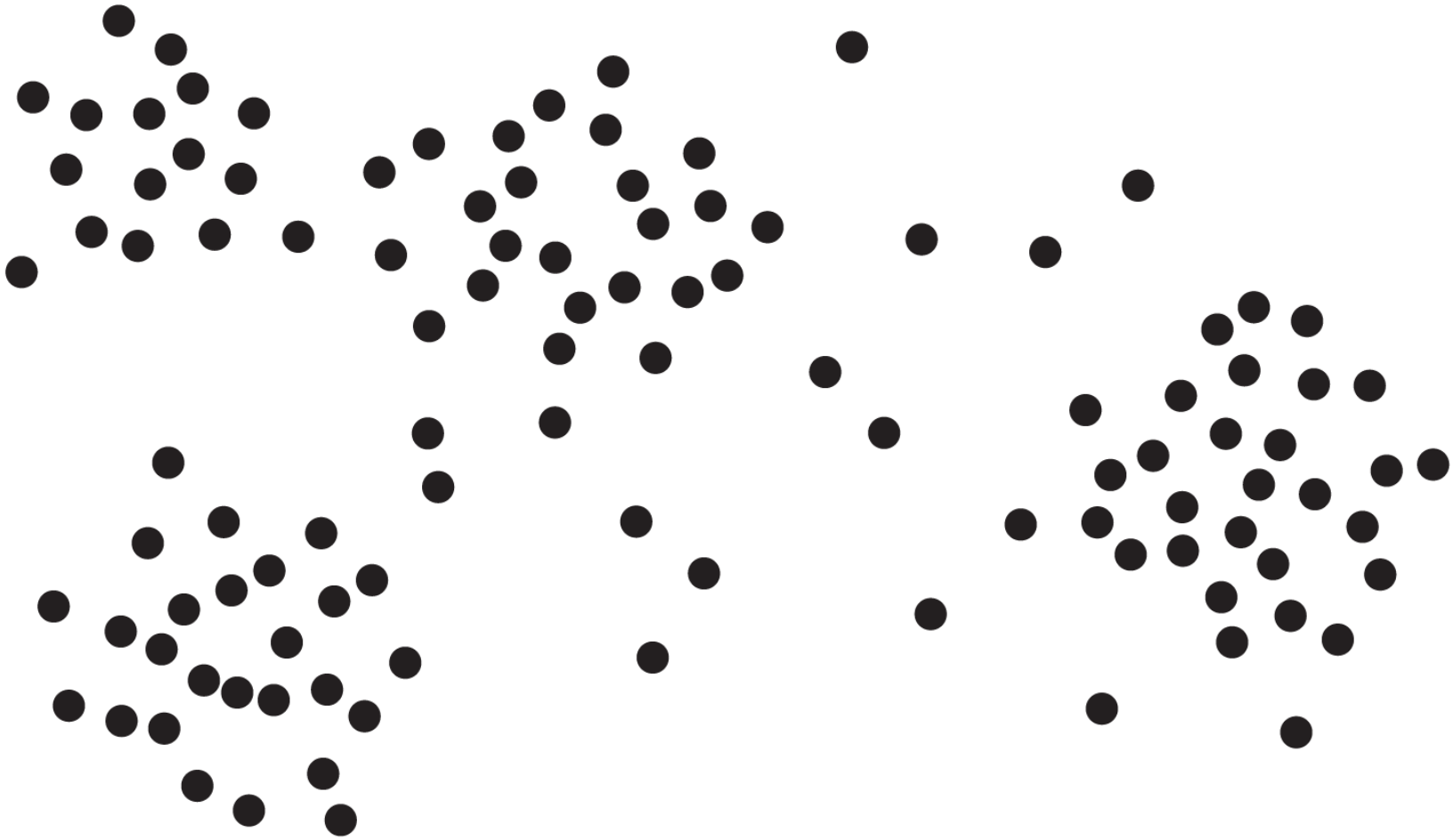
Bad score:
Not very well integrated in the cluster

The Silhouette method



The bootstrap to validate clusters

How do we apply the bootstrap?



We create new datasets by subsampling the original dataset and evaluate how clusters are conserved.

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Some extra notes

No need to invent your own clustering

There are already too many!

https://cran.r-project.org/web/views/Cluster.html

CRAN Task View: Cluster Analysis & Finite Mixture Models

Maintainer: Friedrich Leisch and Bettina Gruen
Contact: Bettina.Gruen at jku.at
Version: 2019-05-29
URL: https://CRAN.R-project.org/view=Cluster

This CRAN Task View contains a list of packages that can be used for finding groups in data and modeling unobserved cross-sectional heterogeneity. Many packages provide functionality for more than one of the topics listed below, the section headings are mainly meant as quick starting points rather than an ultimate categorization. Except for packages stats and cluster (which ship with base R and hence are part of every R installation), each package is listed only once.

Most of the packages listed in this CRAN Task View, but not all are distributed under the GPL. Please have a look at the DESCRIPTION file of each package to check under which license it is distributed.

Hierarchical Clustering:

- Functions `hclust()` from package stats and `agnes()` from cluster are the primary functions for agglomerative hierarchical clustering, function `diana()` can be used for divisive hierarchical clustering. Faster alternatives to `hclust()` are provided by the packages fastcluster and flashClust.
- Function `dendrogram()` from stats and associated methods can be used for improved visualization for cluster dendrograms.
- The dendextend package provides functions for easy visualization (coloring labels and branches, etc.), manipulation (rotating, pruning, etc.) and comparison of dendrograms (tanglegrams with heuristics for optimal branch rotations, and tree correlation measures with bootstrap and permutation tests for significance).
- Package dynamicTreeCut contains methods for detection of clusters in hierarchical clustering dendrograms.
- Package genie implements a fast hierarchical clustering algorithm with a linkage criterion which is a variant of the single linkage method combining it with the Gini inequality measure to robustify the linkage method while retaining computational efficiency to allow for the use of larger data sets.
- hybridHclust implements hybrid hierarchical clustering via mutual clusters.
- Package idendr0 allows to interactively explore hierarchical clustering dendrograms and the clustered data. The data can be visualized (and interacted with) in a built-in heat map, but also in GGobi dynamic interactive graphics (provided by rggobi), or base R plots.
- Package isopam uses an algorithm which is based on the classification of ordination scores from isometric feature mapping. The classification is performed either as a hierarchical, divisive method or as non-hierarchical partitioning.
- The package protoclust implements a form of hierarchical clustering that associates a prototypical element with each interior node of the dendrogram. Using the package's `plot()` function, one can produce dendrograms that are prototype-labeled and are therefore easier to interpret.
- pvclust is a package for assessing the uncertainty in hierarchical cluster analysis. It provides approximately unbiased p-values as well as bootstrap p-values.

Partitioning Clustering:

- Function `kmeans()` from package stats provides several algorithms for computing partitions with respect to Euclidean distance.
- Function `pam()` from package cluster implements partitioning around medoids and can work with arbitrary distances. Function `clara()` is a wrapper to `pam()` for larger data sets. Silhouette plots and spanning ellipses can be used for visualization.
- Package apcluster implements Frey's and Dueck's Affinity Propagation clustering. The algorithms in the package are analogous to the Matlab code published by Frey and Dueck.
- Package ClusterR implements k-means, mini-batch-kmeans, k-medoids, affinity propagation clustering and Gaussian mixture models with the option to plot, validate, predict (new data) and estimate the optimal number of clusters. The package takes advantage of RcppArmadillo to speed up the computationally intensive parts of the functions.
- Package clusterSim allows to search for the optimal clustering procedure for a given dataset.
- Package clustMixType implements Huang's k-prototypes extension of k-means for mixed type data.
- Package evclust implements various clustering algorithms that produce a credal partition, i.e., a set of Dempster-Shafer mass functions representing the membership of objects to clusters.
- Package flexclust provides k-centroid cluster algorithms for arbitrary distance measures, hard competitive learning, neural gas and QT clustering. Neighborhood graphs and image plots of partitions are available for visualization. Some of this functionality is also provided by package cclust.
- Package kernlab provides a weighted kernel version of the k-means algorithm by `kkmeans` and spectral clustering by `specc`.
- Package kml provides k-means clustering specifically for longitudinal (joint) data.
- Package skmeans allows spherical k-Means Clustering, i.e. k-means clustering with cosine similarity. It features several methods, including a genetic and a simple fixed-point algorithm and an interface to the CLUTO vcluster program for clustering high-dimensional datasets.
- Package trimcluster provides trimmed k-means clustering. Package tclust also allows for trimmed k-means clustering. In addition using this package other covariance structures can also be specified for the clusters.

Model-Based Clustering:

- ML estimation:
  - For semi- or partially supervised problems, where for a part of the observations labels are given with certainty or with some probability, package bgmm provides belief-based and soft-label mixture modeling for mixtures of Gaussians with the EM algorithm.
  - EMCluster provides EM algorithms and several efficient initialization methods for model-based clustering of finite mixture Gaussian distribution with unstructured dispersion in unsupervised as well as semi-supervised learning situation.
  - Packages funHDDC and funFEM implement model-based functional data analysis. The funFEM package implements the funFEM algorithm which allows to cluster time series or, more generally, functional data. It is based on a discriminative functional mixture model which allows the clustering of the data in a unique and discriminative functional subspace. This model presents the advantage to be parsimonious and can therefore handle long time series. The funHDDC package impleme
    specific functional subspaces. The funHDDC algorithm is based on a functional mixture model which models and clusters the data into group-specific functional subspaces. The approach allows afterward meaningful interpretations by looking at the group-specific functional curves.
  - Package GLDEX fits mixtures of generalized lambda distributions and for grouped conditional data package mixdist can be used.
  - Package GMCM fits Gaussian mixture copula models for unsupervised clustering and meta-analysis.
  - Package HDclassif provides function `hdde` to fit Gaussian mixture model to high-dimensional data where it is assumed that the data lives in a lower dimension than the original space.
  - Package teigen allows to fit multivariate t-distribution mixture models (with eigen-decomposed covariance structure) from a clustering or classification point of view. Package longclust allows to fit these models as well as Gaussian mixture models to longitudinal data.
  - Package mclust fits mixtures of Gaussians using the EM algorithm. It allows fine control of volume and shape of covariance matrices and agglomerative hierarchical clustering based on maximum likelihood. It provides comprehensive strategies using hierarchical clustering, EM and the Bayesian Information Criterion (BIC) for clustering, density estimation, and discriminant analysis. Package Rmixmod provides tools for fitting mixture models of multivariate Gaussian or multinomial con
    analysis point of view. Package mclust as well as packages mixture and Rmixmod provide all 14 possible variance-covariance structures based on the eigenvalue decomposition.
  - Package MetabolAnalyze fits mixtures of probabilistic principal component analysis with the EM algorithm.
  - For grouped conditional data package mixdist can be used.
  - Package MixAll provides EM estimation of diagonal Gaussian, gamma, Poisson and categorical mixtures combined based on the conditional independence assumption using different EM variants and allowing for missing observations. The package accesses the clustering part of the Statistical ToolKit STK++.
  - mixtools provides fitting with the EM algorithm for parametric and non-parametric (multivariate) mixtures. Parametric mixtures include mixtures of multinomials, multivariate normals, normals with repeated measures, Poisson regressions and Gaussian regressions (with random effects). Non-parametric mixtures include the univariate semi-parametric case where symmetry is imposed for identifiability and multivariate non-parametric mixtures with conditional independent assumption. In
    available.
    Fitting finite mixtures of uni- and multivariate scale mixtures of skew-normal distributions with the EM algorithm is provided by package mixsmsn.
  - Package MoEClust fits parsimonious finite multivariate Gaussian mixtures of experts models via the EM algorithm. Covariates may influence the mixing proportions and/or component densities and all 14 constrained covariance parameterizations from package mclust are implemented.
  - Package movMF fits finite mixtures of von Mises-Fisher distributions with the EM algorithm.
  - mritc provides tools for classification using normal mixture models and (higher resolution) hidden Markov normal mixture models fitted by various methods.
  - prabclus clusters a presence-absence matrix object by calculating an MDS from the distances, and applying maximum likelihood Gaussian mixtures clustering to the MDS points.
  - Package psychomix estimates mixtures of the dichotomous Rasch model (via conditional ML) and the Bradley-Terry model. Package mixRasch estimates mixture Rasch models, including the dichotomous Rasch model, the rating scale model, and the partial credit model with joint maximum likelihood estimation.
  - Package pmclust allows to use unsupervised model-based clustering for high dimensional (ultra) large data. The package uses pbdMPI to perform a parallel version of the EM algorithm for mixtures of Gaussians.
  - Package rebmix implements the REBMIX algorithm to fit mixtures of conditionally independent normal, lognormal, Weibull, gamma, binomial, Poisson, Dirac or von Mises component densities as well as mixtures of multivariate normal component densities with unrestricted variance-covariance matrices.
- Bayesian estimation:
  - Bayesian estimation of finite mixtures of multivariate Gaussians is possible using package bayesm. The package provides functionality for sampling from such a mixture as well as estimating the model using Gibbs sampling. Additional functionality for analyzing the MCMC chains is available for averaging the moments over MCMC draws, for determining the marginal densities, for clustering observations and for plotting the uni- and bivariate marginal densities.
  - Package bayesmix provides Bayesian estimation using JAGS.
  - Package bclust allows Bayesian clustering using a spike-and-slab hierarchical model and is suitable for clustering high-dimensional data.
  - Package Bmix provides Bayesian Sampling for stick-breaking mixtures.
  - Package bmixture provides Bayesian estimation of finite mixtures of univariate Gamma and normal distributions.
  - Package dpmixsim fits Dirichlet process mixture models using conjugate models with normal structure. Package profdpm determines the maximum posterior estimate for product partition models where the Dirichlet process mixture is a specific case in the class.
  - Package GSM fits mixtures of gamma distributions.
  - Package IMIFA fits Infinite Mixtures of Infinite Factor Analyzers and a flexible suite of related models for clustering high-dimensional data. The number of clusters and/or number of cluster-specific latent factors can be non-parametrically inferred, without recourse to model selection criteria.
  - Package mcclust implements methods for processing a sample of (hard) clusterings, e.g. the MCMC output of a Bayesian clustering model. Among them are methods that find a single best clustering to represent the sample, which are based on the posterior similarity matrix or a relabeling algorithm.
  - Package mixAK contains a mixture of statistical methods including the MCMC methods to analyze normal mixtures with possibly censored data.
  - Package PReMiuM is a package for profile regression, which is a Dirichlet process Bayesian clustering where the response is linked non-parametrically to the covariate profile.
  - Package rjags provides an interface to the JAGS MCMC library which includes a module for mixture modelling.
- Other estimation methods:
  - Package AdMit allows to fit an adaptive mixture of Student-t distributions to approximate a target density through its kernel function.
  - Package CEC uses cross-entropy clustering to automatically remove unnecessary clusters, while at the same time allowing the simultaneous use of various types of Gaussian mixture models.
  - Circular and orthogonal regression clustering using redescending M-estimators is provided by package edci.

Other Cluster Algorithms:

- Package ADPclust allows to cluster high-dimensional data based on a two dimensional decision plot. This density-distance plot plots for each data point the local density against the shortest distance to all observations with a higher local density value. The cluster centroids of this non-iterative procedure can be selected using an interactive or automatic selection mode.
- Package amap provides alternative implementations of k-means and agglomerative hierarchical clustering.
- Package biclust provides several algorithms to find biclusters in two-dimensional data.
- Package cba implements clustering techniques for business analytics like "rock" and "proximus".
- Package CHsharp clusters 3-dimensional data into their local modes based on a convergent form of Choi and Hall's (1999) data sharpening method.
- Package clue implements ensemble methods for both hierarchical and partitioning cluster methods.
- Package CoClust implements a cluster algorithm that is based on copula functions and therefore allows to group observations according to the multivariate dependence structure of the generating process without any assumptions on the margins.
- Fuzzy clustering and bagged clustering are available in package e1071. Further and more extensive tools for fuzzy clustering are available in package fclust.
- Package compHclust provides complimentary hierarchical clustering which was especially designed for microarray data to uncover structures present in the data that arise from 'weak' genes.
- Package dbscan provides a fast reimplementation of the DBSCAN (density-based spatial clustering of applications with noise) algorithm using a kd-tree.

If you have VERY large datasets

Computing the distance between each pair of datapoint is NOT reasonable

Function `clara()` is a wrapper to `pam()` for larger data sets.

Partitioning method (k-medoids)

It uses **subsamples** of the original dataset,
looks for **conserved clusters**,
Finds their **medoids**,
Compute the **distance of the remaining data points** to the medoids,
Attribute a cluster to each datapoint.

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

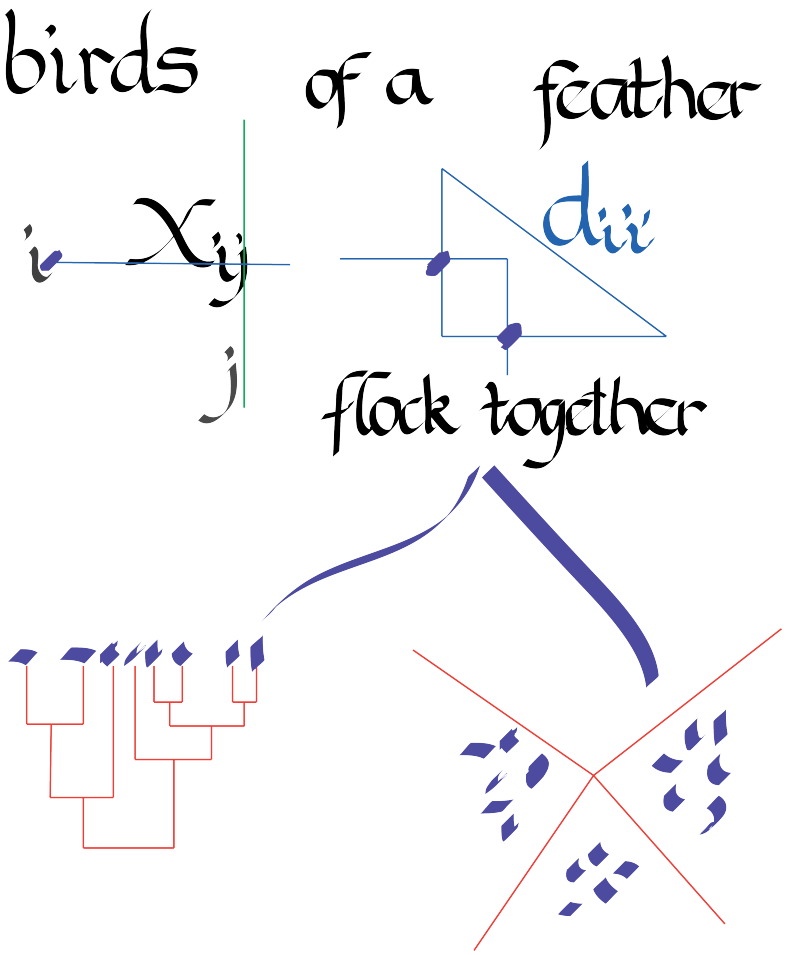
2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Summary

Summary: the clustering workflow



1. Start with the data.
   What are the **measurements**?
   What **type** of data?

2. Define or select a **metric (distance)** to evaluate the (dis)similarity between 2 samples.

3. Choose a **clustering method**: bottom-up (*hierarchical*) or top-down (*k-methods*)

4. **Validate** the clustering and evaluate the **optimal number** of clusters

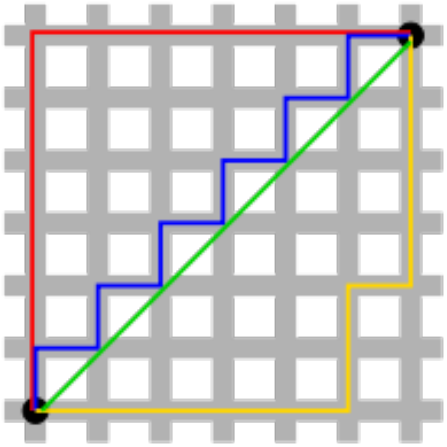
5. **Augment** your data with the cluster information

Distances
We saw at the start of the Lecture how finding the **right** distance is an essential first step in a clustering analysis. This is a case when the ***garbage in, garbage out*** motto is in full force. Always choose a distance which is **scientifically meaningful** and compare output from as many distances as possible. Sometimes the same data require **different distances** when **different scientific objectives** are pursued.

Partitioning and Aggregating
We saw two different types of clustering approaches:
iterative partitioning approaches such as kmeans and kmedoids (PAM) that alternated between estimating the clusters and assigning points to them and **hierarchical clustering** approaches that agglomerate points and then small clusters into larger ones in a nested sequence of sets that can be represented by hierarchical clustering trees.

Cluster validation
Clustering algorithms ***always*** deliver clusters so we need to assess their quality and the number of clusters to choose carefully.
These validation steps are done using visualization tools and repeating the clustering on many resamples of the data. We saw how statistics such as the bss/wss or log(wss) can be calibrated using simulation on data where we understand the group structure and can provides useful benchmarks for choosing the number of cluster.

The choice or definition of **distance depends on the data**



Euclidean distance (L2) $d(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_p - b_p)^2}.$

Manhattan distance (L1) $d(A,B) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_p - b_p|.$

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

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

Edit (Hamming) distance

Binary distance

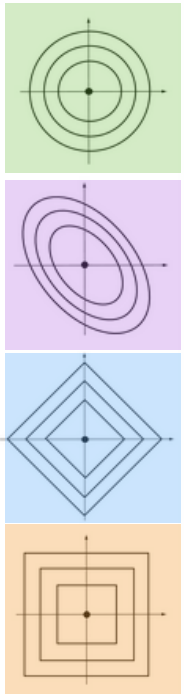
Jaccard distance $d_J(S,T) = 1 - J(S,T) = \frac{f_{01} + f_{10}}{f_{01} + f_{10} + f_{11}}.$

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

Weighted Euclidean distance

Mahalanobis distance

…



Questions

Goals for this lecture

1. Review measures of **(dis)similarity** that help us define clusters.

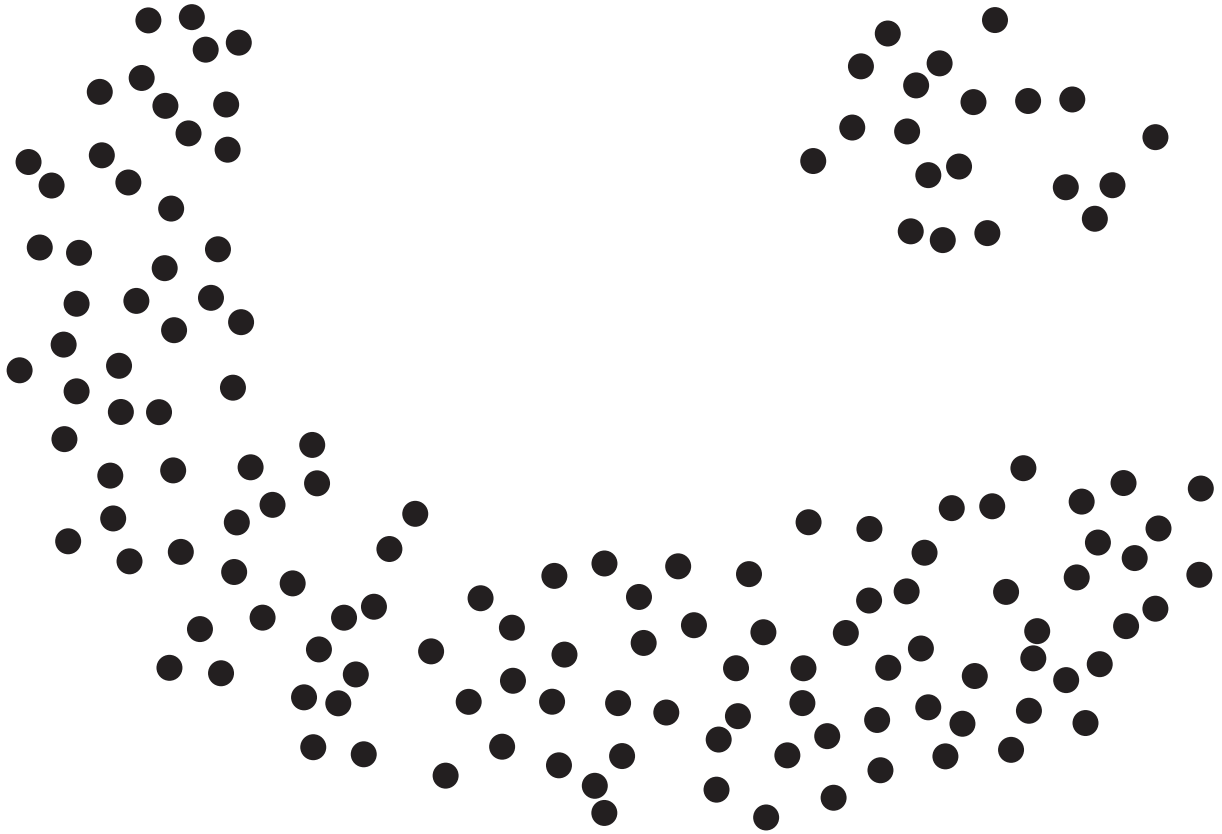
2. Explain **non-parametric methods** such as ***k-means*** or ***k-medoids***

3. Explain the **recursive approach** to clustering that combines observations and groups into a hierarchy of sets and called ***hierarchical clustering***.

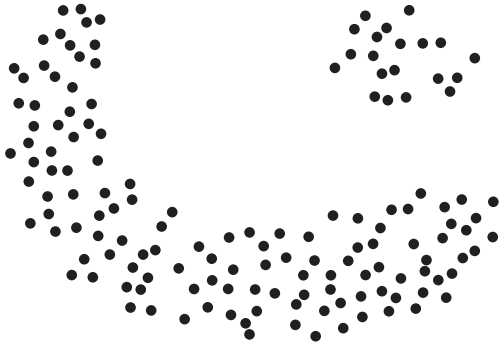
4. Understand how to **validate the clusters** and select the **optimal number of clusters**.

Some more extras

Density-based clustering



Density-based clustering



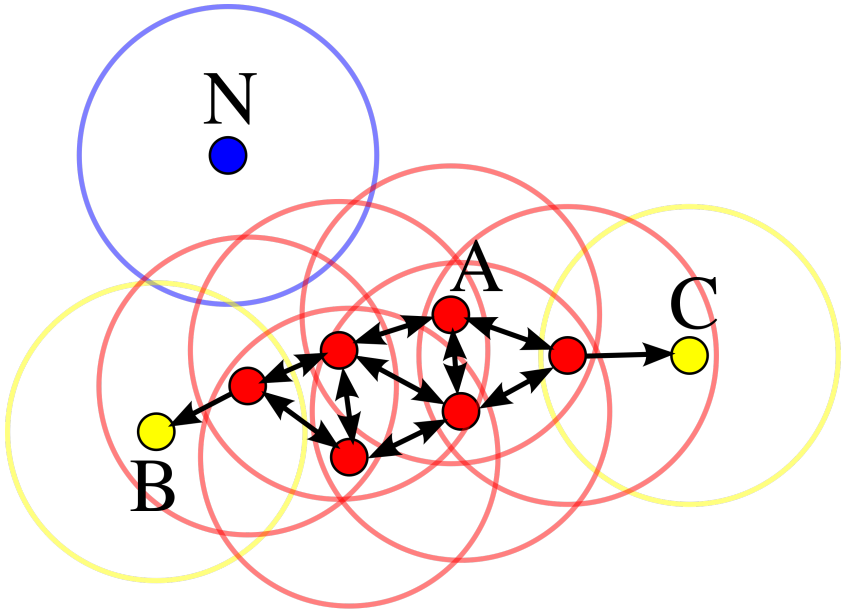
2 parameters:

Eps (ε) : the maximum distance between 2 points so that they are considered as “reachable” from one another

minPts: the minimum number of points for a cluster to be created.

It classifies each data-point into

- core points
- directly reachable points
- noise / outliers



Algorithm: **DBSCAN**
Density-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise

Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu,1996

https://en.wikipedia.org/wiki/DBSCAN