DMI

Unsupervised Learning:
Cluster Analysis

Data Mining and Data integration in Biomedicine
Master in Bioinformatics

Janet Piñero
Medbioinformatics Solutions SL
2025-2026

Outline

DMI

Introduction

Data Preprocessing

Most popular clustering methods
- Hierarchical Clustering
- K-Means

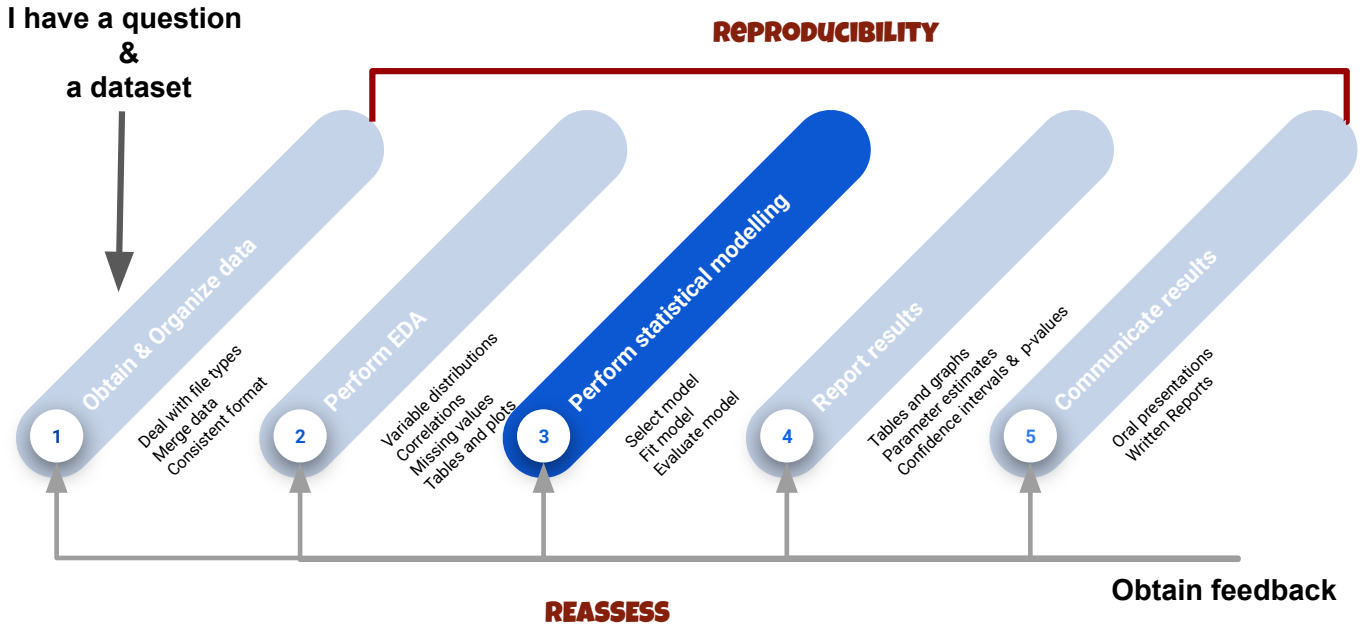
Similarity Measures for Clustering Results

Hands-on session

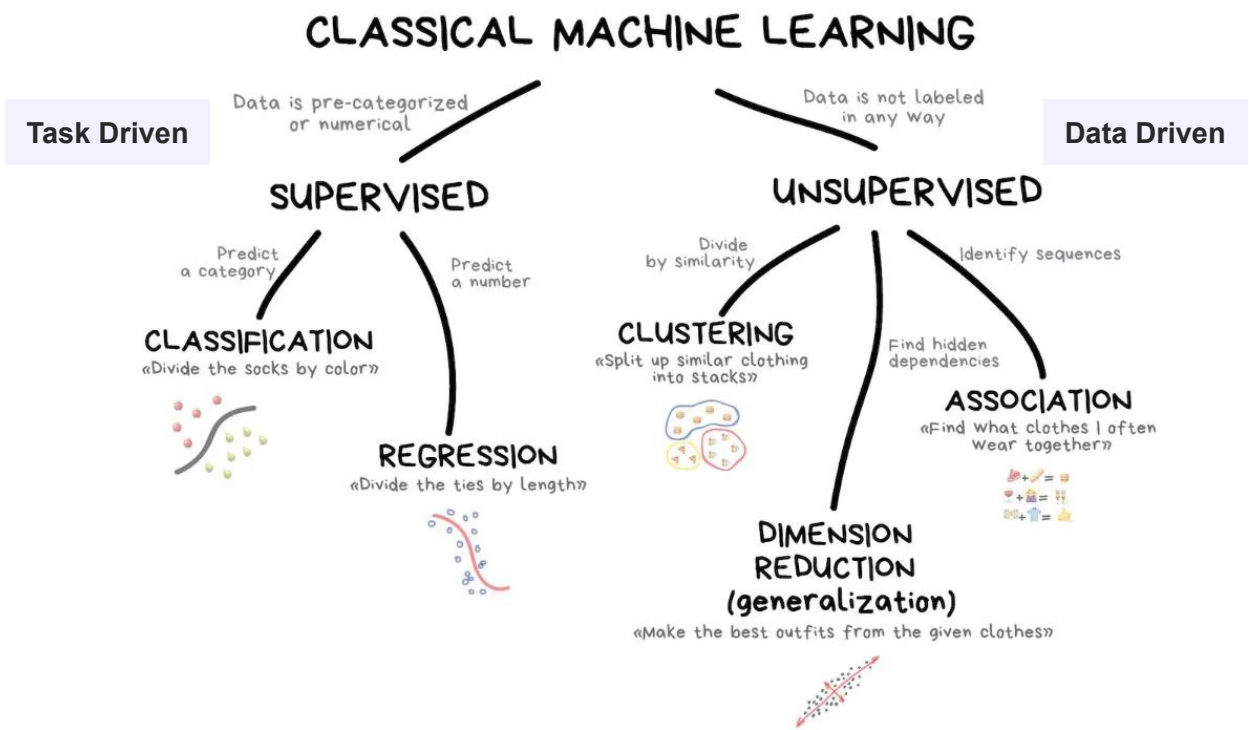
References

Recap from previous lectures

DMI



Modified from https://x.com/siminaboca/status/1298870717291917312

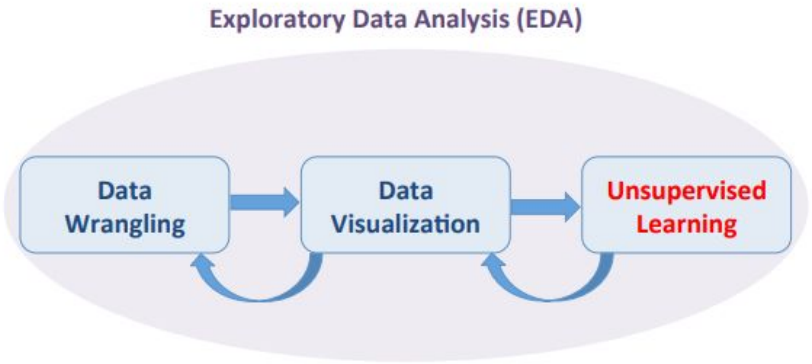


Objective: **Predictive Models** **Pattern/Structure Recognition**

Why Unsupervised Learning?

DMI

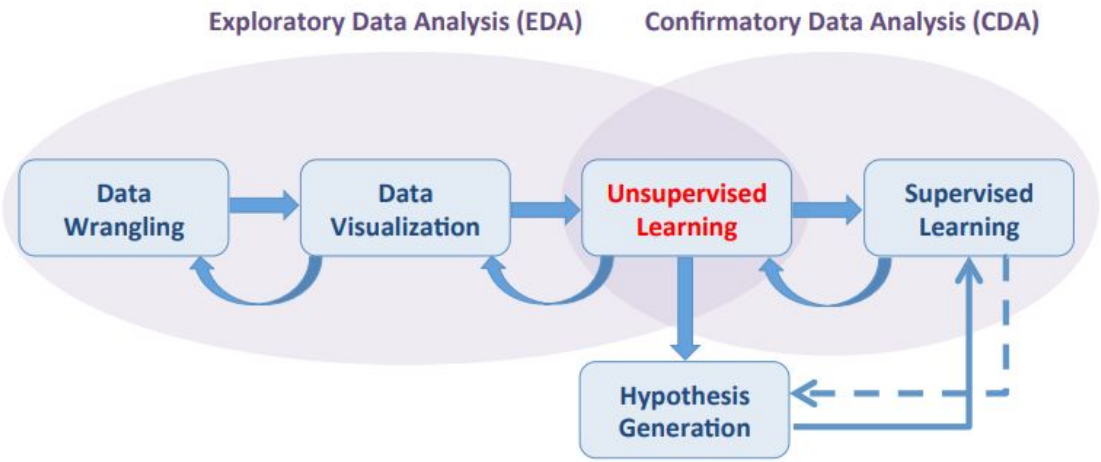
Unsupervised learning is an important part of the data analysis pipeline.



Why Unsupervised Learning?

DMI

Unsupervised learning is an important part of the data analysis pipeline.



Classes of Unsupervised Learning

DMI

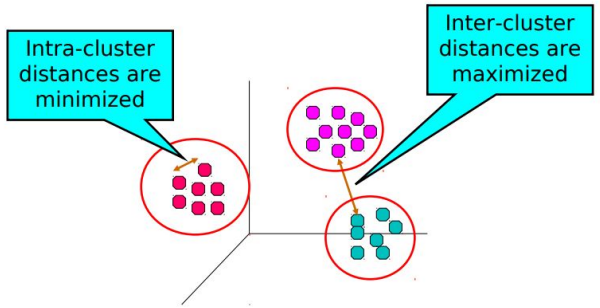
Four main classes of methods

- **Clustering**
- **Dimension Reduction**
- Association Rule Mining
- Network Analysis

What is Clustering?

DMI

- Clustering is the classification of data objects into similarity groups (clusters) according to a defined distance measure.
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



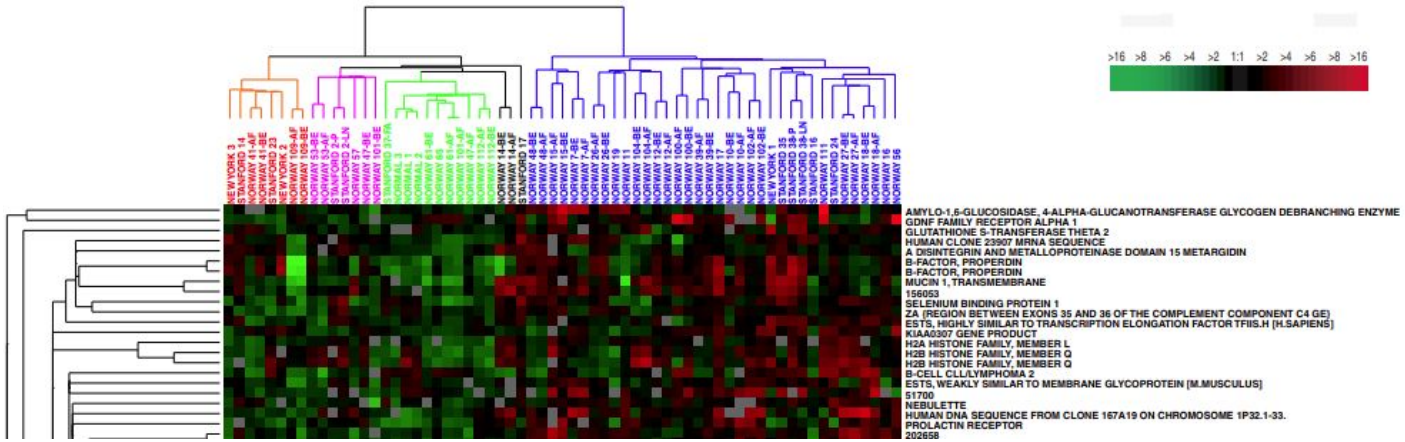
Applications of clustering

DMI

- Understanding
  - Group related documents for browsing, group genes and proteins that have similar functionality, or group of similar patients
- Summarization
  - Reduce the size of large data sets

Clustering is widely used in Biomedical Data analysis

Molecular portraits of human breast tumours
https://doi.org/10.1038/35021093

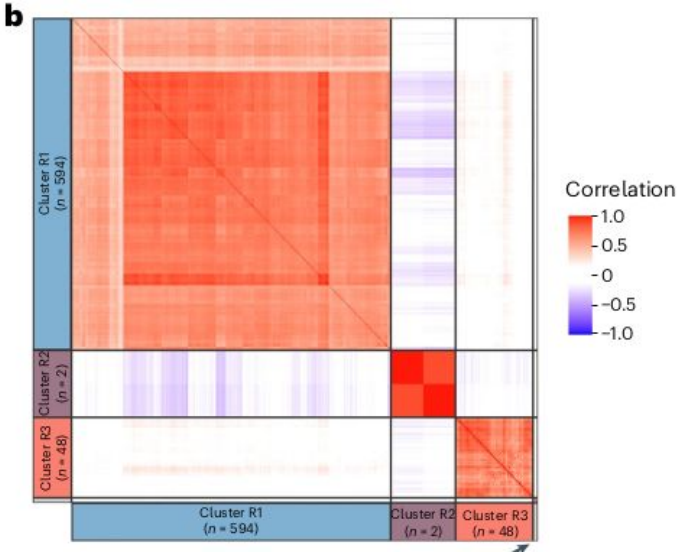


The tumours could be classified into subtypes distinguished by pervasive differences in their gene expression patterns.

Clustering is widely used in Biomedical Data analysis

Five latent factors underlie response to immunotherapy

https://doi.org/10.1038/s41588-024-01899-0



The Notion of a Cluster can be Ambiguous

How many clusters?

The Notion of a Cluster can be Ambiguous

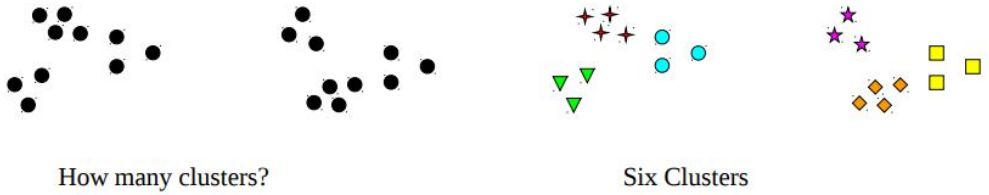
DMI

How many clusters?

Two Clusters

The Notion of a Cluster can be Ambiguous

DMI



How many clusters?

Six Clusters

Two Clusters

Four Clusters

What to Cluster?

DMI

- observations: clustering samples to find cancer subtypes (most common)
- features: clustering genes based on similar functions (pathways)
- or both (bi-clustering): Clustering features is similar to clustering observations.

Data Preparation

DMI

- Data matrix in which rows are observations (individuals, samples, genes), and columns are variables or features (for example, treatments,)
- Any missing value in the data must be removed or estimated.
- The data must be standardized (i.e., scaled) to make variables comparable.

Data Transformations

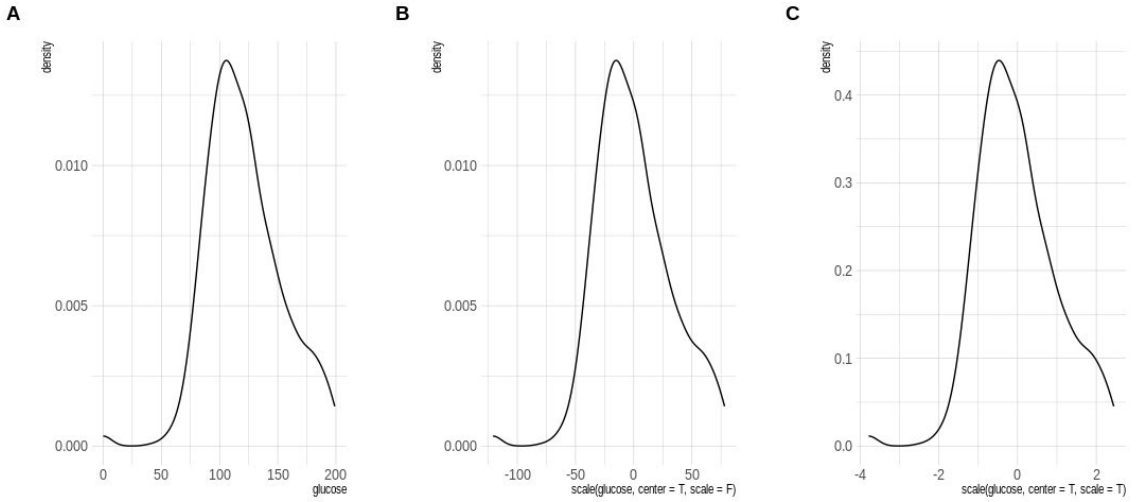
DMI

Choice depends on data set!

- Center & standardize
  i. Center: subtract from each value the mean of the corresponding vector
  ii. Standardize: divide centered columns by standard deviation
  ⇒ Mean = 0 and STDEV = 1
  iii. Center & scale with the `scale()` function in R
- Log transformation (right skewed data, scale reduction)
- Rank transformation: replace measured values by ranks
- No transformation

Scaling data

Technique for comparing data that isn't measured in the same way.



$$x_{scaled} = (x - \bar{x}) / s$$

Where: x: real x-value, $\bar{x}$: Sample mean, s: Sample Standard Deviation

This is also known as data standardization, and it basically involves converting each original value into a z-score.

Can we find things that are close together?

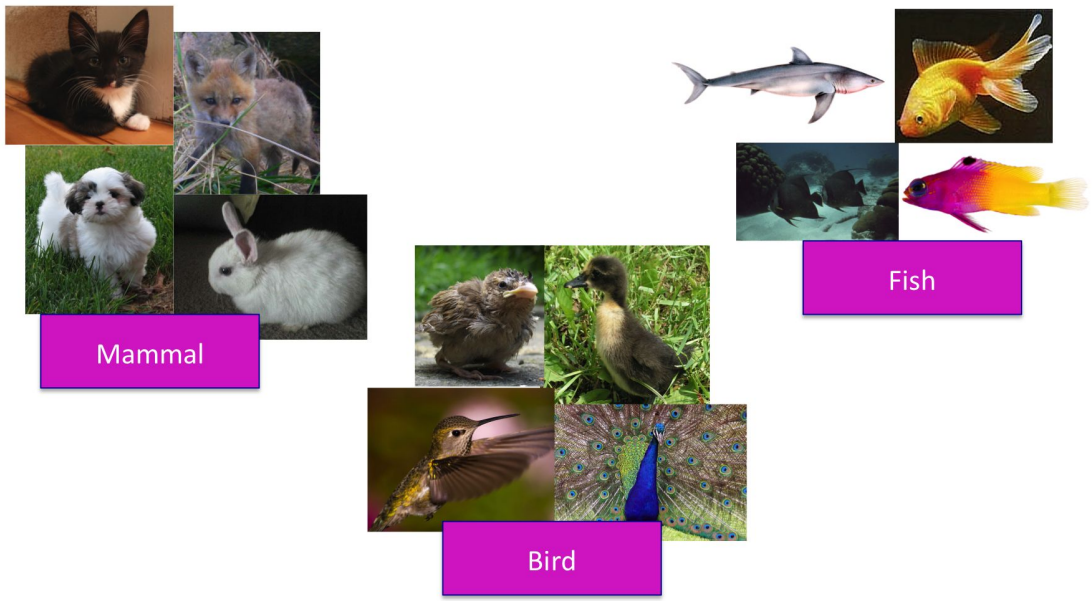
DMI

Clustering organizes things that are close into groups

- How do we define close?
- How do we group things?
- How do we visualize the grouping?
- How do we interpret the grouping?

Can we find things that are close together?

DMI



How do we define close: Distance Methods

DMI

- Euclidean distance for two profiles X and Y $d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

in a p-dimensional space (for p features) is the square root of the sum of squares of the differences in all p coordinate directions

How do we define close: Distance Methods

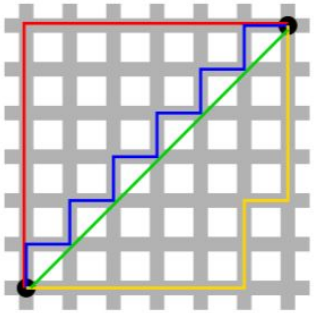
DMI

- Euclidean distance for two profiles X and Y

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

  Disadvantages: not scale invariant, not for negative correlations,sensitive to outliers

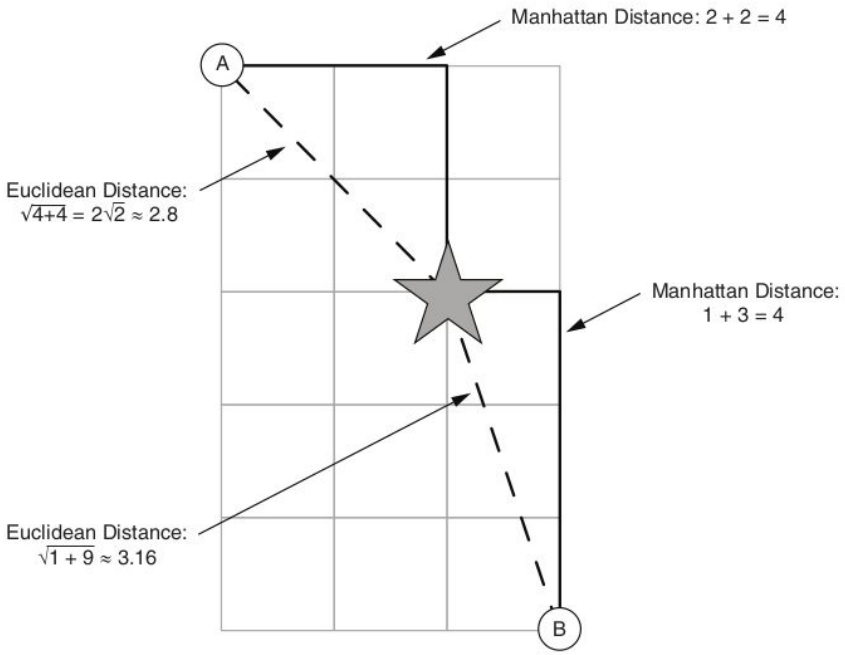
- Maximum, Manhattan, Canberra, binary, Minkowski, ...



Manhattan The Manhattan, City Block, Taxicab distance takes the sum of the absolute differences in all coordinates.

$$d_{man}(x,y) = \sum_{i=1}^{n}|(x_i - y_i)|$$

How do we define close: Distance Methods



How do we define close: Distance Methods

DMI

- Euclidean distance for two profiles X and Y

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

  Disadvantages: not scale invariant, not for negative correlations,sensitive to outliers

- Maximum, Manhattan, Canberra, binary, Minkowski, ...
- Correlation-based distance: *1 - r*
  i. Pearson correlation coefficient (PCC)

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{(\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)(\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2)}}$$

     Disadvantage: outlier sensitive

  ii. Spearman correlation coefficient (SCC)

     Similar to PCC but with ranked values!

There Are Many More Distance Measures

DMI

- If the distances among items are quantifiable, then clustering is possible.
- Choose the most accurate and meaningful distance measure for a given application.
- If uncertain then choose several distance measures and compare the results.

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144059

Computations related to distances in R

The **dist** function in R computes one of six choices of distance (euclidean, maximum, manhattan, canberra, binary, minkowski).

The function returns a special object of class dist that encodes the relevant vector of size $n \times (n-1)/2$

```
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
```

Distance Matrices in R

DMI

Euclidean distance matrix

```
dist(y[1:4,], method = "euclidean")
##          g1       g2       g3
## g2 4.793697
## g3 4.932658 6.354978
## g4 4.033789 4.788508 1.671968
```

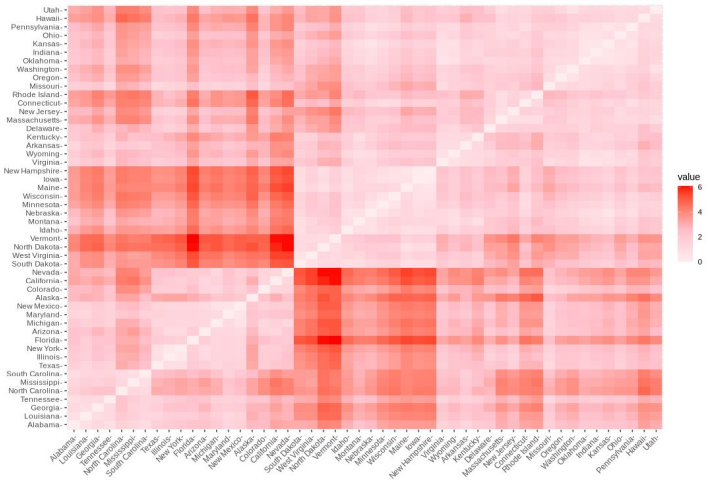
	t1	t2	t3	t4	t5
g1	1.427263848	-1.60013297	-0.27322387	-0.35976940	-1.09318615
g2	-0.027209166	0.17379160	-1.05988238	0.25003624	-0.73992414
g3	1.344959565	0.50414740	0.20156162	-1.56317869	1.30343029
g4	-0.788519956	-0.12394346	-0.13044004	-0.98351818	-0.72065724
g5	-1.290287996	-0.84442002	-0.34766912	0.71512620	0.24259925
g6	-0.531449212	-0.49860548	-1.46322280	-0.25035108	-0.04514773
g7	-0.168676086	0.46947280	0.50400094	-0.70399753	-0.75442242
g8	0.666282625	-0.06089866	-1.05612297	1.86070512	1.22968305
g9	0.899917923	-0.43259950	-0.31228205	-1.07448298	-0.88534990
g10	-0.777449825	0.91749389	-1.09835081	-2.44370401	-0.74535657

Correlation-based distance matrix

Correlation matrix

```
c <- cor(t(y), method="pearson")
as.matrix(c)[1:4,1:4]
##             g1         g2          g3         g4
## g1  1.00000000 -0.2965885 -0.00206139 -0.4042011
## g2 -0.29658847  1.0000000 -0.91661118 -0.4512912
## g3 -0.00206139 -0.9166112  1.00000000  0.7435892
## g4 -0.40420112 -0.4512912  0.74358925  1.0000000
```

Distance Matrices in R



Within R it is simple to compute and visualize the distance matrix using the functions **get_dist()** and **fviz_dist()** from the factoextra R package.

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6

	Murder	Assault	UrbanPop	Rape
Alabama	1.24256408	0.78283935	-0.52090661	-0.003416473
Alaska	0.50786248	1.10682252	-1.21176419	2.484202941
Arizona	0.07163341	1.47880321	0.99898006	1.042878388
Arkansas	0.23234938	0.23086801	-1.07359268	-0.184916602
California	0.27826823	1.26281442	1.75892340	2.067820292

```
df <- USArrests
df <- na.omit(df)
df <- scale(df)
distance <- get_dist(df)
fviz_dist(distance, gradient = list(low = "white", mid ="pink", high = "red"))
```

Different Types of Clusterings

DMI

- **Hierarchical versus Partitional:** whether the set of clusters is nested or unnested. Partitional clustering is simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one cluster. If we permit clusters to have subclusters, then we obtain a hierarchical clustering, which is a set of nested clusters that are organized as a tree
- **Exclusive versus Overlapping versus Fuzzy:** *Exclusive* clustering assign each object to a single cluster. An *overlapping* or non-exclusive clustering is used to reflect the fact that an object can simultaneously belong to more than one group (class). In a *fuzzy* clustering every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs).
- **Complete versus Partial :**A complete clustering assigns every object to a cluster, whereas a partial clustering does not.

Clustering Algorithms

DMI

- Hierarchical methods
  - **Hierarchical clustering**
  - Biclustering
  - Convex clustering & convex biclustering
- Partition-based methods
  - **K-means clustering**
  - Model-based soft clustering
  - Spectral clustering
- Density-based methods
  - **DBSCAN**

Hierarchical Clustering Algorithm

DMI

Hierarchical clustering gives a sequence of nested clusters, organized in a hierarchical tree structure, called the dendrogram.

Two classes of algorithms:

- Bottom-Up (Agglomerative): Start from n individual clusters, and group them together into using a measure of similarity.
- Top-Down (Divisive): Start from one cluster containing all objects, and break them down using a measure of distance.

Note that agglomerative clustering is good at identifying small clusters. Divisive hierarchical clustering is good at identifying large clusters.

Hierarchical Clustering Approaches in R

DMI

1. Agglomerative approach (bottom-up)

   `hclust` [in stats package] and `agnes` [in cluster package]

2. Divisive approach (top-down)

   `diana` [in cluster package]

We will focus on bottom-up clustering.

Hierarchical Clustering Algorithm

DMI

Initialization

- Assign each item to its own cluster.

Iteration

- Determine the clusters with the smallest distance. If there are several equidistant choices then pick one randomly.
- Join these closest clusters to a new (larger) cluster.
- Place a node for new cluster at the height corresponding to a predefined distance rule between the joined clusters.
- Update distances by computing the distance of the new cluster to all other clusters.

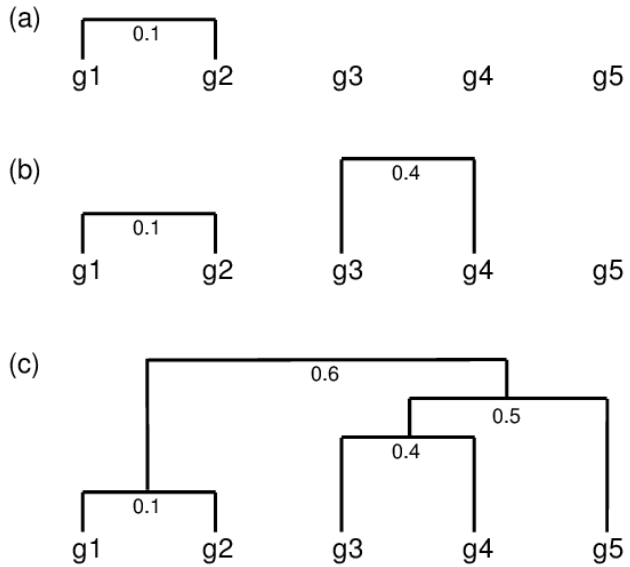
Termination

- When only two clusters remain place the root at the height according to the predefined distance rule.

Hierarchical Clustering

DMI

Agglomerative Approach



How to make a dendrogram?

DMI

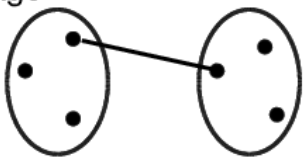
- Need a measure of similarity/distance between observations
  - many choices

- Need a measure of similarity/distance between clusters (i.e. sets of observations), aka linkage function

Cluster Linkage

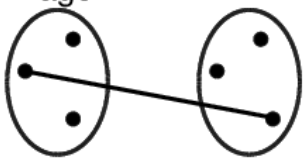
DMI

Single Linkage



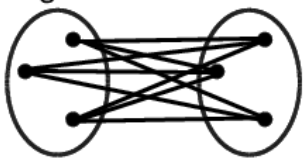
single linkage: minimum distance of points in the two clusters

Complete Linkage



complete linkage: maximum distance of points in the two clusters

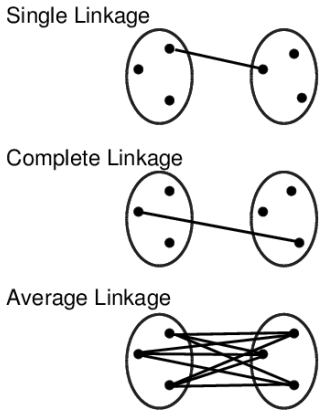
Average Linkage



average linkage: average distance of points in the two clusters

Cluster Linkage

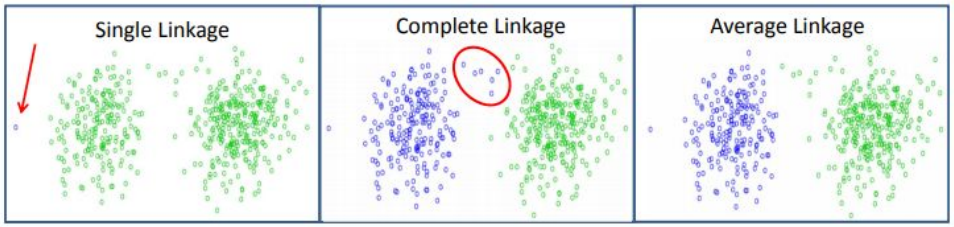
DMI



Outliers

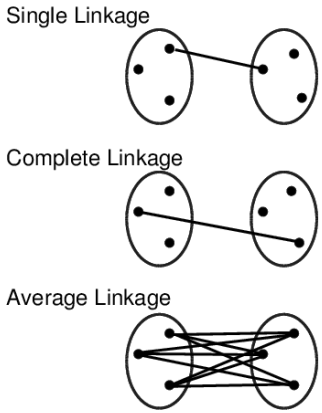
Which Linkage Function?

- Single linkage: Works with diverse shapes, but very sensitive to outliers/noise
- Complete linkage: Gives comparable cluster sizes, and is robust to outliers, but works better with spherical distributions
- Average linkage: A compromise that is relatively robust to outliers, but also works better with spherical distributions



Cluster Linkage

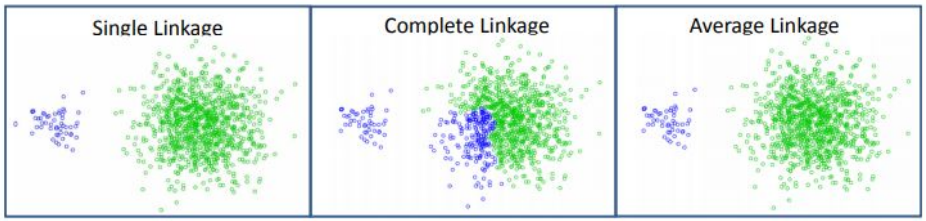
DMI



Unbalanced clusters

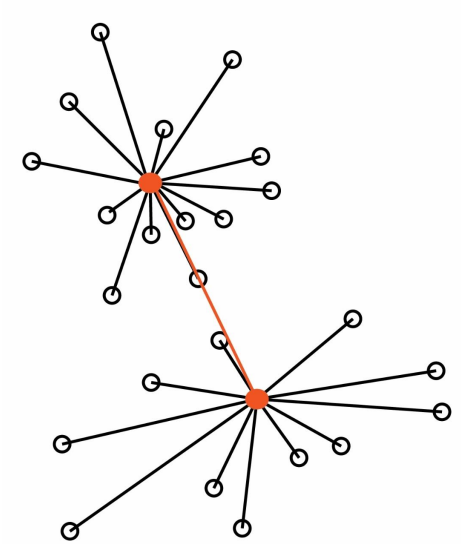
Which Linkage Function?

- Single linkage: Works with diverse shapes, but very sensitive to outliers/noise
- Complete linkage: Gives comparable cluster sizes, and is robust to outliers, but works better with spherical distributions
- Average linkage: A compromise that is relatively robust to outliers, but also works better with spherical distributions



Cluster Linkage

DMI

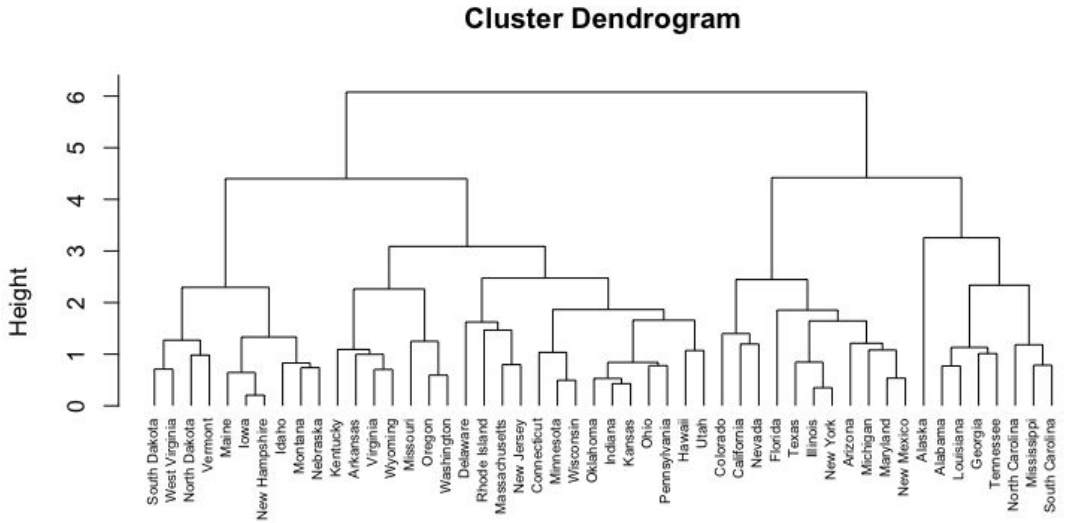


Ward's minimum variance method: It minimizes the total within-cluster variance.
At each step the pair of clusters with minimum between-cluster distance are merged.

See more information here

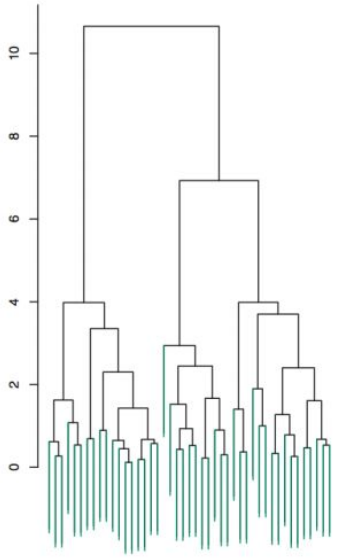
Interpreting the dendrogram

DMI



Interpreting the dendrogram

- At the bottom of the tree, there is a leaf for each observation.
- As we move up the tree, leaves fuse into branches, depending on how similar observations are.
- The earlier (lower in the tree) fusions occur, the more similar the observations are to each other.
- Observations that fuse later (near the top of the tree) can be quite different.
- The height (on the vertical axis) where branches for two observations are first fused indicates how different they are from each other
- The horizontal axis is not informative: cannot conclude how similar two observations are from their proximity along the horizontal axis

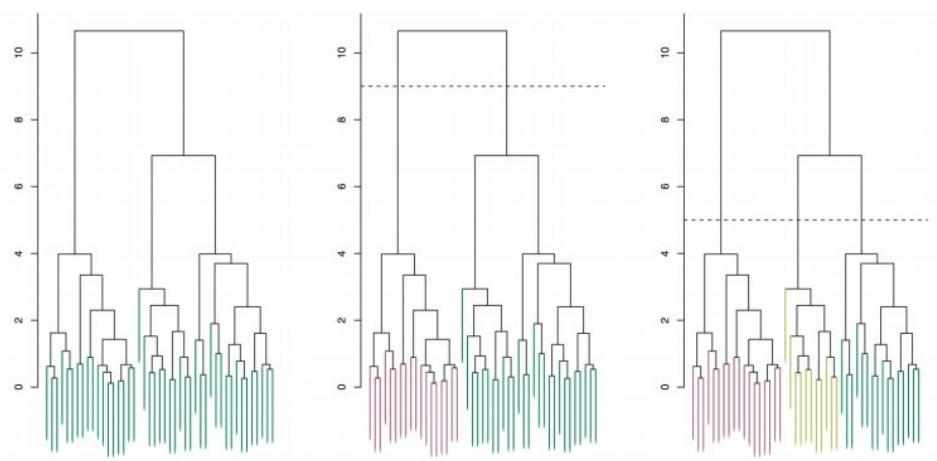


Intro Stat Learning, James et al (2011)

Choosing the Number of Clusters

DMI

Generally difficult: By examining the dendrogram along the vertical axis

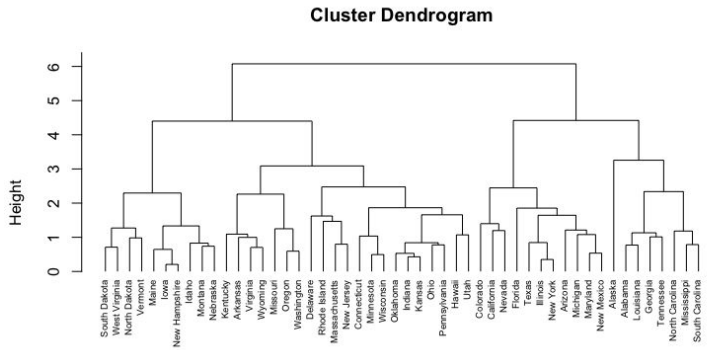


Tree Cutting to Obtain Discrete Clusters

DMI

Cut tree by:

1. Node height in tree
2. Number of clusters



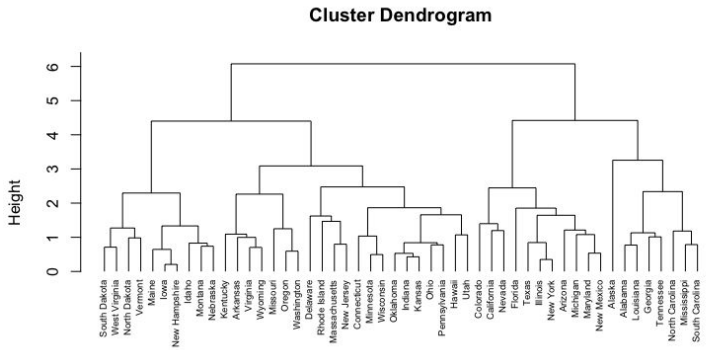
Hierarchical Clustering Approaches in R

DMI

```
# Dissimilarity matrix
d <- dist(df, method = "euclidean")

# Hierarchical clustering using Complete Linkage
hc1 <- hclust(d, method = "complete" )

# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)
```



Hierarchical Clustering Approaches in R

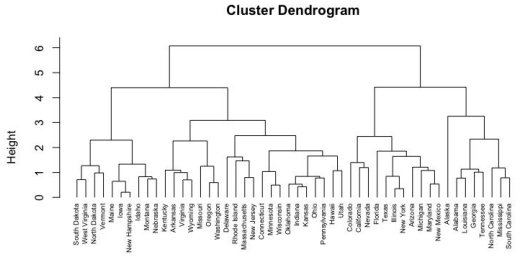
The height of the cut to the dendrogram controls the number of clusters obtained. It plays the same role as the k in k-means clustering. In order to identify sub-groups (i.e. clusters), we can cut the dendrogram with `cutree`:

```
hc1 <- hclust(d, method = "complete" )


# Cut tree into 4 groups
sub_grp <- cutree(hc1, k = 4)


# Number of members in each cluster
table(sub_grp)
```

```
sub_grp
 1  2  3  4
 8 11 21 10
```




```
> sub_grp
    Alabama      Alaska     Arizona    Arkansas  California
          1           1           2           3           2
Connecticut    Delaware     Florida     Georgia      Hawaii
          3           3           2           1           3
   Illinois     Indiana        Iowa      Kansas    Kentucky
          2           3           4           3           3
      Maine    Maryland Massachusetts    Michigan   Minnesota
          4           2           3           2           3
```

Hierarchical Clustering Approaches in R

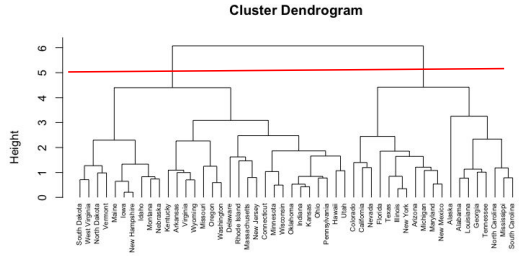
The height of the cut to the dendrogram controls the number of clusters obtained. It plays the same role as the k in k-means clustering. In order to identify sub-groups (i.e. clusters), we can cut the dendrogram with `cutree`:

```
hc1 <- hclust(d, method = "complete" )


# Cut tree into 4 groups
sub_grp <- cutree(hc1, h = 5)


# Number of members in each cluster
table(sub_grp)
```

```
> table(sub_grp)
sub_grp
 1  2
19 31
>
```




```
> sub_grp
     Alabama       Alaska      Arizona     Arkansas
           1            1            1            2
 Connecticut     Delaware      Florida      Georgia
           2            2            1            1
    Illinois      Indiana         Iowa       Kansas
           1            2            2            2
       Maine     Maryland Massachusetts     Michigan
           2            1            2            1
```

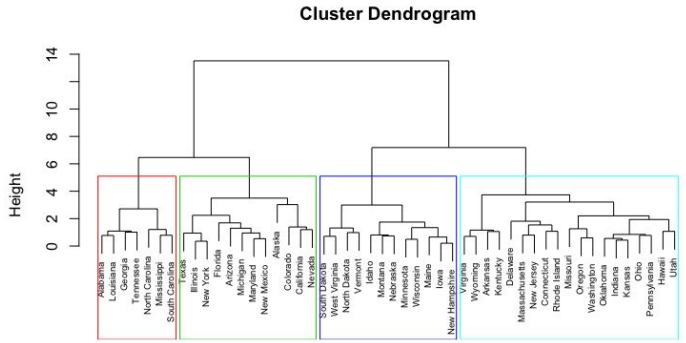
Hierarchical Clustering Approaches in R

DMI

It's also possible to draw the dendrogram with a border around the 4 clusters. The argument border is used to specify the border colors for the rectangles:

```
plot(hc1, cex = 0.6)
rect.hclust(hc1, k = 4, border = 2:5)
```

Border: R color codes 2, 3, 4, 5



Hierarchical Clustering

DMI

Advantages:

- Gives a family of nested clusters
- Computationally efficient
- Works in high dimensions

Disadvantages:

- No optimization criterion
- Final solution chosen by the data analyst
- Different merging (splitting) criteria give different solutions

K-Means Clustering

DMI

Finds the best partition of the observations into a priori defined number of clusters K.

- Simple and intuitive objective
- Computationally efficient, and can be applied to datasets with a large number of observations
- No hierarchy among clusters, if K is changed, the cluster memberships will also change

K-Means Clustering

DMI

(1) Choose the number of k clusters

(2) Randomly assign items to the k clusters

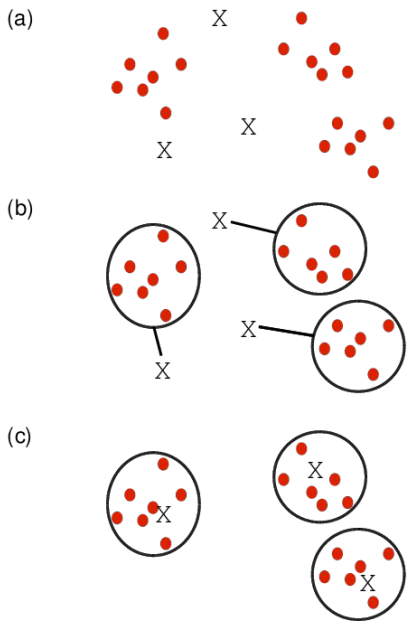
(3) Calculate new centroid for each of the k clusters

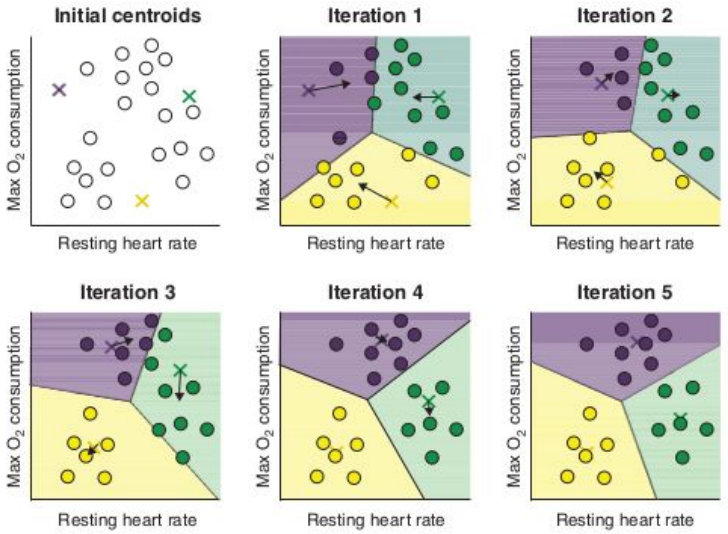
(4) Calculate the distance of all items to the k centroids

(5) Assign items to closest centroid

(6) Repeat until clusters assignments are stable

K-Means Clustering





https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-Means Clustering

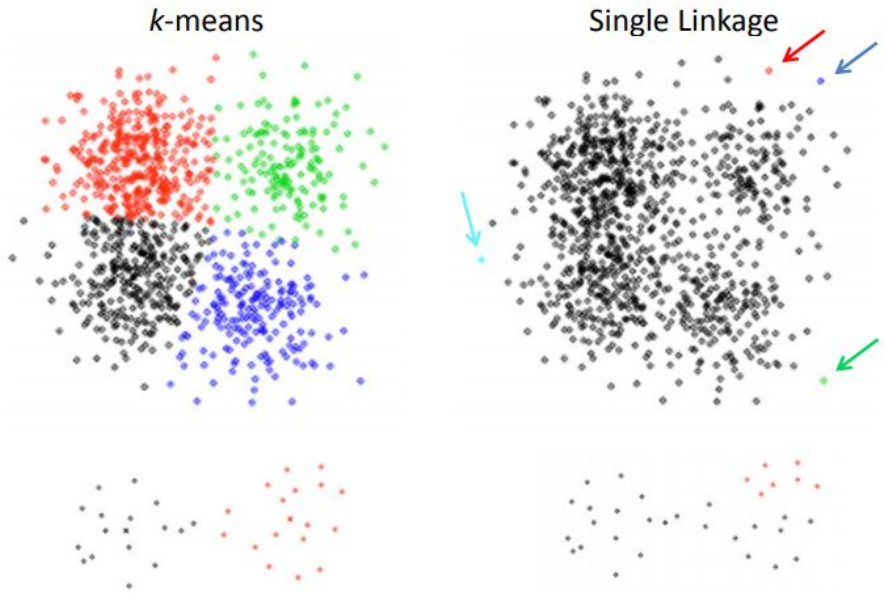
DMI

- Initial centroids are often chosen randomly.
    - Clusters produced vary from one run to another.
- The centroid is the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance
- K-means will converge (points stop changing assignment) typically in the first few iterations.
    - Often the stopping condition is changed to 'Until relatively few points change clusters'

K-Means Clustering

DMI

Examples where K-means works well



K-Means Clustering

DMI

- In K-means there is no hierarchy among clusters, if K is changed, the cluster memberships will also change.
- IK-means works best with compact spherical clusters with comparable number of members.
- K-means clustering is a very simple and fast algorithm and it can efficiently deal with very large data sets.
- One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters.
- An additional disadvantage of K-means is that it’s sensitive to outliers
- May result in small/empty clusters (potentially due to outliers)

K-Means in R

DMI

```
k2 <- kmeans(df, centers = 2, iter.max=10, nstart = 25)
```

the number of times the algorithm is run before results are returned.

multiple initial configurations, that is, number of initial random centroids. Only the best one is returned

the number of clusters

It is strongly recommended to compute k-means clustering with a large value of nstart such as 25 or 50, in order to have a more stable result

This means that R will try 25 (or 50) different random starting assignments and then select the best results corresponding to the one with the lowest within cluster variation (**total within-cluster sum of squares**).

Evaluating K-means Clusters

DMI

- Most common measure is Sum of Squared Error (SSE) or Within-Cluster Sum of Squares
  - For each point, the error is the distance to the nearest cluster center

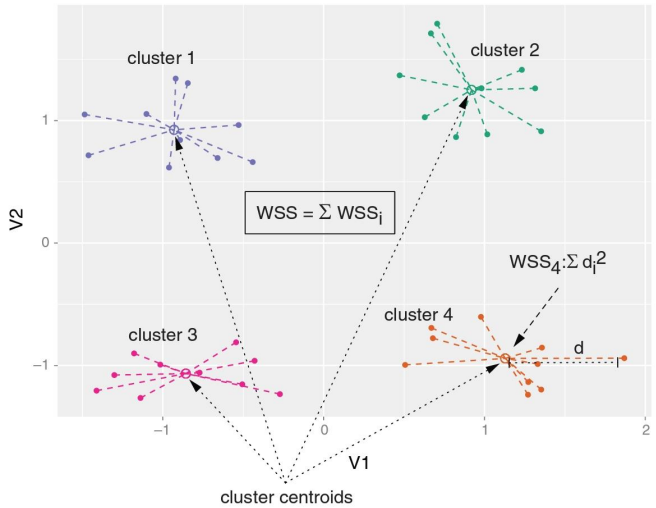
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - m_i\|^2$$

  - x is a data point in cluster Ci , mi is the center for cluster Ci as the mean of all points in the cluster and ||.|| is the Euclidean distance.
  - Given two clusterings, we can choose the one with the smallest error
  - **Only compare clusterings with the same K**! One easy way to reduce SSE is to increase K, the number of clusters
- K-Means is a heuristic to minimize SSE.

Heuristic: A heuristic is a shortcut or rule of thumb used to solve problems quickly when exact solutions are impractical.

Evaluating K-means Clusters

DMI



K-Means in R

DMI

```
k2 <- kmeans(df, centers = 2, iter.max=10, nstart = 25)
```

the number of times the algorithm is run before results are returned.

multiple initial configurations, that is, number of initial random centroids. Only the best one is returned

The output of `kmeans` is a list with several bits of information. The most important being:

- `cluster`: A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
- `centers`: A matrix of cluster centers.
- `totss`: The total sum of squares.
- `withinss`: Vector of within-cluster sum of squares, one component per cluster.
- `tot.withinss`: Total within-cluster sum of squares, i.e. sum(withinss).
- `betweenss`: The between-cluster sum of squares, i.e. totss-tot.withinss.
- `size`: The number of points in each cluster.

The total within-cluster sum of square measures the compactness (i.e goodness) of the clustering and we want it to be as small as possible.

The between clusters sum of squares. In fact it is the mean of distances between cluster centers.

K-Means in R

DMI

```
k2 <- kmeans(df, centers = 2, iter.max=10, nstart = 25)
```

The output of `kmeans` is a list with several bits of information. The most important being:

`cluster` — Cluster assignment

A vector indicating **which cluster each observation belongs to**.

- Length = number of data points
- Values = `1, 2, ..., K`

Interpretation:
"Observation *i* is assigned to cluster `cluster k`."

`centers` — Cluster centroids

A matrix giving the **centroid (mean vector)** of each cluster.

- Rows = clusters
- Columns = variables
- Each row is the **average of all points in that cluster**

Interpretation:
"The typical (average) observation in each cluster."

K-Means in R

DMI

```
k2 <- kmeans(df, centers = 2, iter.max=10, nstart = 25)
```

`totss` — Total sum of squares $\text{totss} = \sum_x \|x - \bar{x}\|^2$

The **total variability in the data before clustering**.

- Measures overall spread of the dataset
- **Does not depend on K**

Interpretation:
"How spread out the data are overall."

`withinss` — Within-cluster sum of squares (per cluster)

A vector measuring **how compact each cluster is**.

- One value per cluster
- Smaller = tighter cluster

$\text{withinss}_i = \sum_{x \in C_i} \|x - m_i\|^2$

Interpretation:
"How close points are to their own centroid."

`tot.withinss` — Total within-cluster sum of squares (SSE)

The **total clustering error** minimized by k-means.

- Sum of all within-cluster errors
  Always decreases as K increases

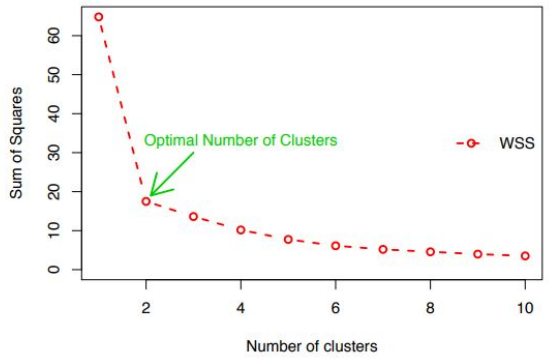
$\text{tot.withinss} = \sum_i \text{withinss}_i$

Interpretation:
"How well the clustering fits the data."

Choosing the number of clusters:

K-means clustering:

- Elbow method
- Silhouette method
- Gap statistic



Choosing the number of clusters: Elbow Method

DMI

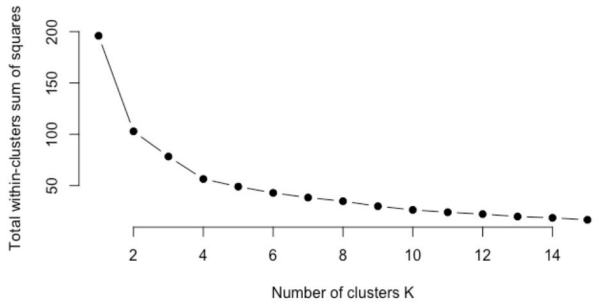
The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible. Thus, we can use the following algorithm to define the optimal clusters:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of *k*. For instance, by varying *k* from 1 to 10 clusters
2. For each *k*, calculate the total within-cluster sum of square (wss)
3. Plot the curve of wss according to the number of clusters *k*.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

Choosing the number of clusters: Elbow Method

DMI

```
# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(df, k, nstart = 10 )$tot.withinss
}
# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15
# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```

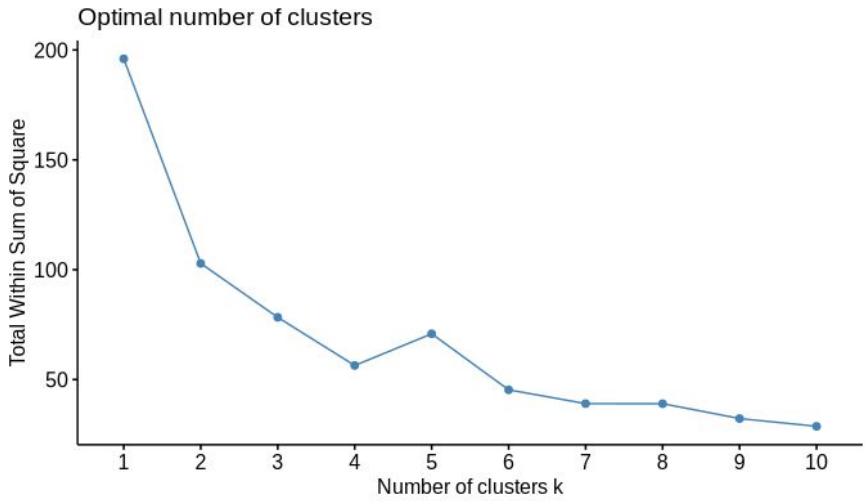


Choosing the number of clusters: Elbow Method

DMI

library(factoextra)

```
fviz_nbclust(df, kmeans, method = "wss")
```

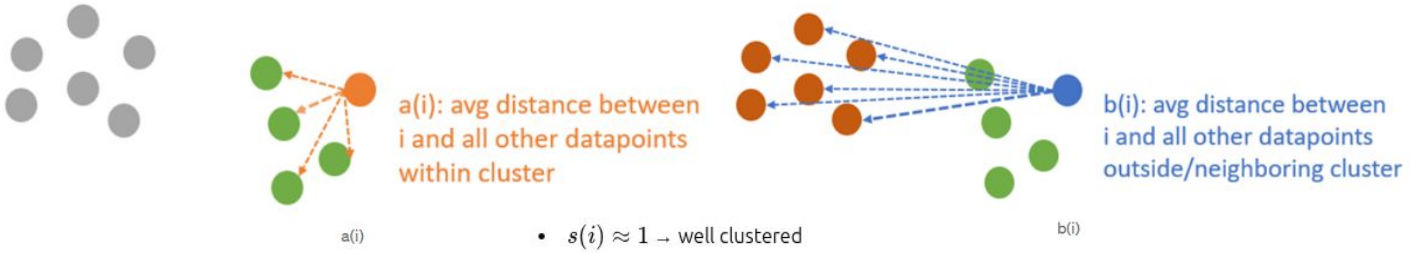


Choosing the number of clusters: silhouette Method

`silhouette() computes or extracts silhouette information (cluster)`

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

- S(i) is the silhouette coefficient of the data point i.
- a(i) is the average distance between i and all the other data points in the cluster to which i belongs.
- b(i) is the average distance from i to all clusters to which i does not belong.

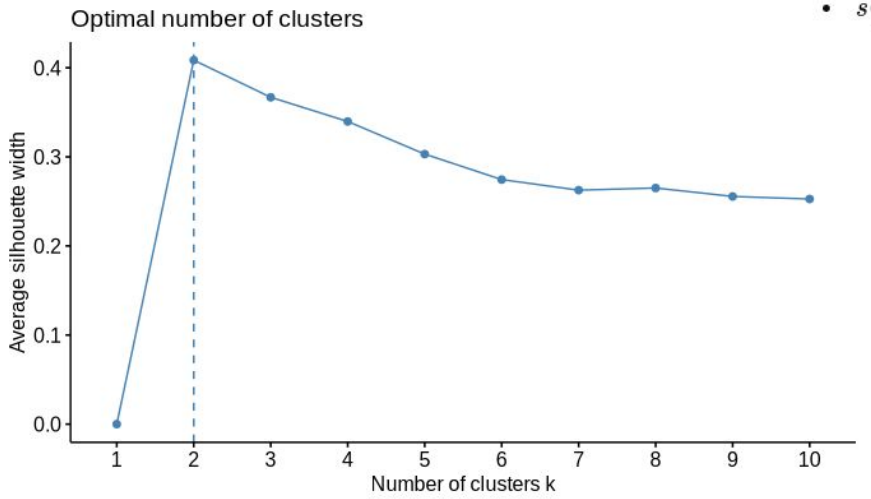


- $s(i) \approx 1 \rightarrow$ well clustered
- $s(i) \approx 0 \rightarrow$ on the boundary
- $s(i) < 0 \rightarrow$ likely misclassified

Choosing the number of clusters: silhouette Method

library(factoextra)

```
fviz_nbclust(df, kmeans, method = "silhouette")
```



- $s(i) \approx 1 \rightarrow$ well clustered
- $s(i) \approx 0 \rightarrow$ on the boundary
- $s(i) < 0 \rightarrow$ likely misclassified

Other methods for Cluster Validation in R

DMI

- cluster.stats() compute several cluster validity statistics from a clustering and a dissimilarity matrix (fpc)
- clValid() calculate validation measures for a given set of clustering algorithms and number of clusters (clValid)
- clustIndex() calculate the values of several clustering indexes, which can be independently used to determine the number of clusters existing in a data set (cclust)
- NbClust() provide 30 indices for cluster validation and determining the number of clusters (NbClust)

K means vs hierarchical clustering

DMI

The advantage of using hierarchical clustering over k means is, it doesn't require advanced knowledge of number of clusters. However, some of the advantages which k means has over hierarchical clustering are as follows:

- It uses less memory.
- It converges faster.
- Unlike hierarchical, k means doesn't get trapped in mistakes made on a previous level. It improves iteratively.
- k means is non-deterministic in nature, i.e.. after every time you initialize, it will produce different clusters. On the contrary, hierarchical clustering is deterministic.

Note: K means is preferred when the data is numeric. Hierarchical clustering is preferred when the data is categorical.

Small Decisions with Big Consequences

DMI

- Should the observations or features first be standardized in some way?
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
  - Where should we cut the dendrogram in order to obtain clusters?
- In the case of K-means clustering, how many clusters should we look for in the data?

In practice, we try several different choices, and look for the one with the most useful or interpretable solution. With these methods, there is no single right answer—any solution that exposes some interesting aspects of the data should be considered.

Bootstrap evaluation of the clusters

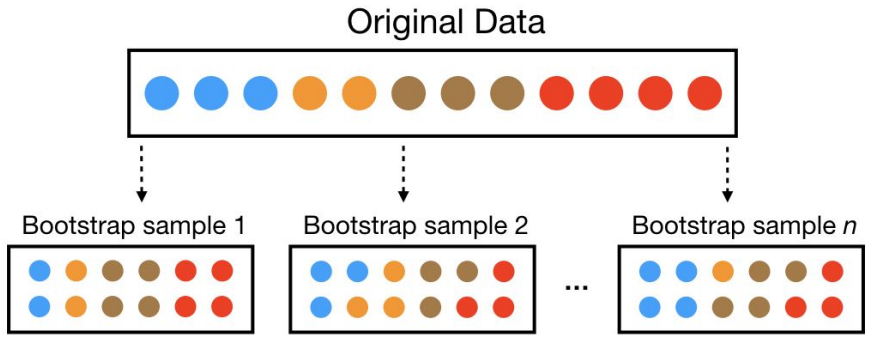
DMI

1. Cluster the data as usual.
2. Draw a new dataset (of the same size as the original) by resampling the original dataset with replacement (meaning that some of the data points may show up more than once, and others not at all). Cluster the new dataset.
3. For every cluster in the original clustering, find the most similar cluster in the new clustering (the one that gives the maximum Jaccard coefficient) and record that value. If this maximum Jaccard coefficient is less than 0.5, the original cluster is considered to be dissolved—it didn't show up in the new clustering.
4. A cluster that's dissolved too often is probably not a "real" cluster.
5. Repeat steps 2–3 several times.

https://search.r-project.org/CRAN/refmans/fpc/html/clusterboot.html

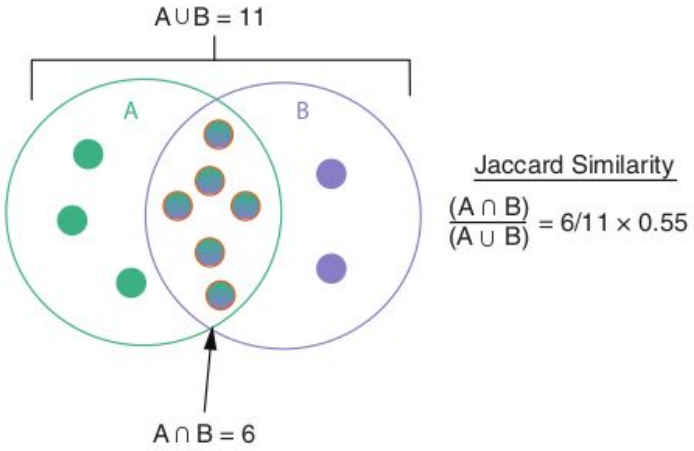
Bootstrap evaluation of the clusters

Since samples are drawn with replacement, each bootstrap sample is likely to contain duplicate values. In fact, on average, ≈63.21 % of the original sample ends up in any particular bootstrap sample



Bootstrap evaluation of the clusters

DMI



Bootstrap evaluation of the clusters

DMI

1. Cluster the data as usual.
2. Draw a new dataset (of the same size as the original) by resampling the original dataset with replacement (meaning that some of the data points may show up more than once, and others not at all). Cluster the new dataset.
3. For every cluster in the original clustering, find the most similar cluster in the new clustering (the one that gives the maximum Jaccard coefficient) and record that value. If this maximum Jaccard coefficient is less than 0.5, the original cluster is considered to be dissolved—it didn't show up in the new clustering.
4. A cluster that's dissolved too often is probably not a "real" cluster.
5. Repeat steps 2–3 several times.

https://search.r-project.org/CRAN/refmans/fpc/html/clusterboot.html

Bibliography

The Elements of Statistical Learning. Hastie, Tibshirani and Friedman

Avoiding common pitfalls when clustering biological data (2016) DOI: 10.1126/scisignal.aad1932

Hands-on session

DMI

Go to this page and accept the assignment

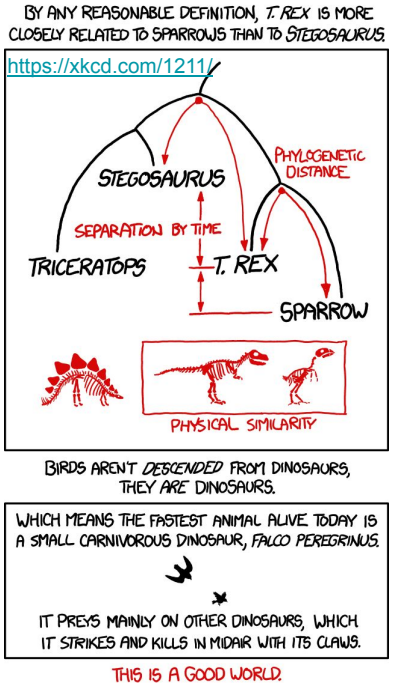
Some materials

http://uc-r.github.io/kmeans_clustering

http://uc-r.github.io/hc_clustering

Deadline to submit the assignment: 15/03/2026

Cophenetic distance



An example of computing the cophenetic distance

In the clustering of biological information such as data from microarray experiments, the **cophenetic distance** of two objects is a measure of how similar those two objects have to be in order to be grouped into the same cluster. The cophenetic distance between two objects is the height of the dendrogram where the two branches that include the two objects merge into a single branch