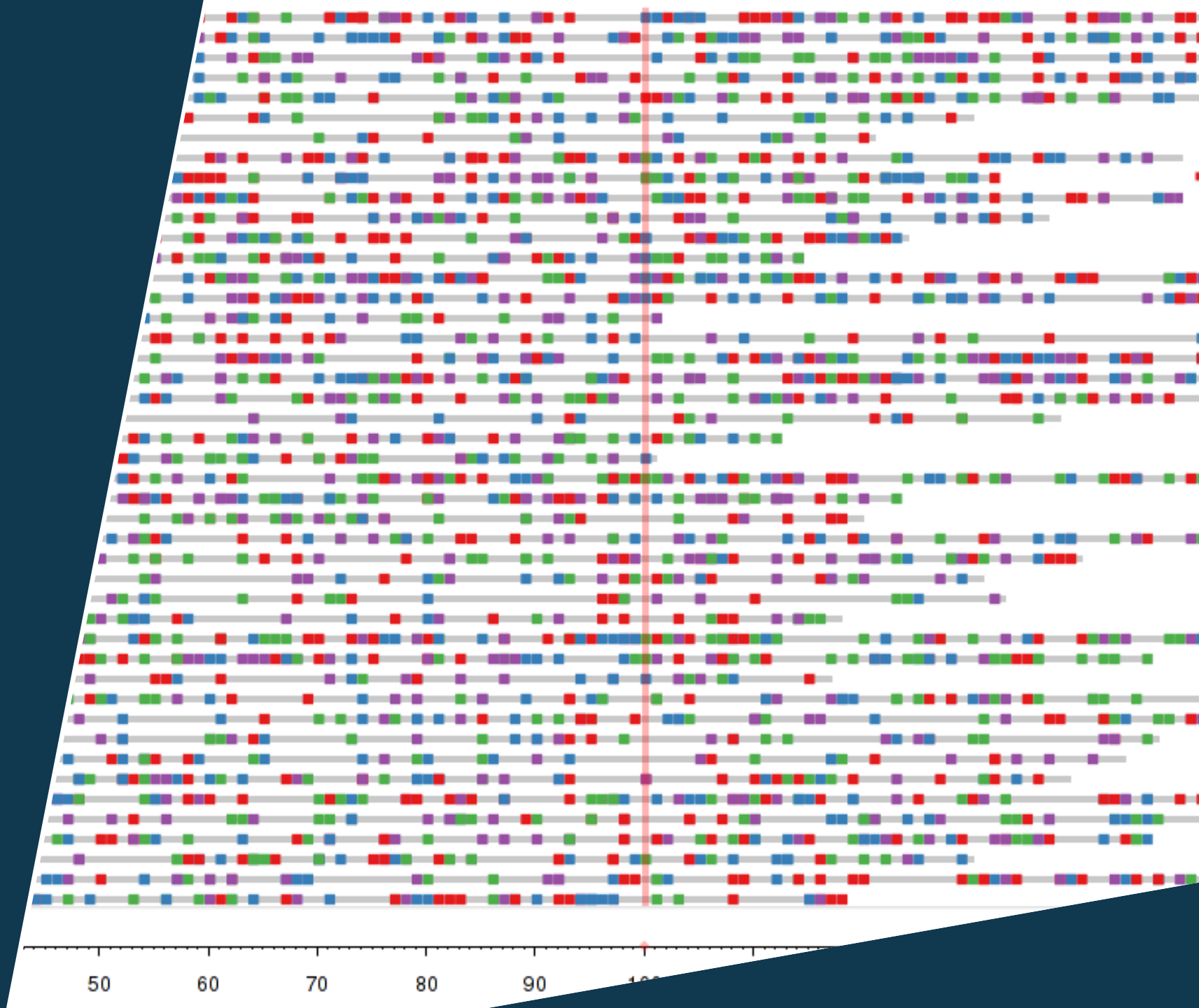
Seriation in

Megaplots



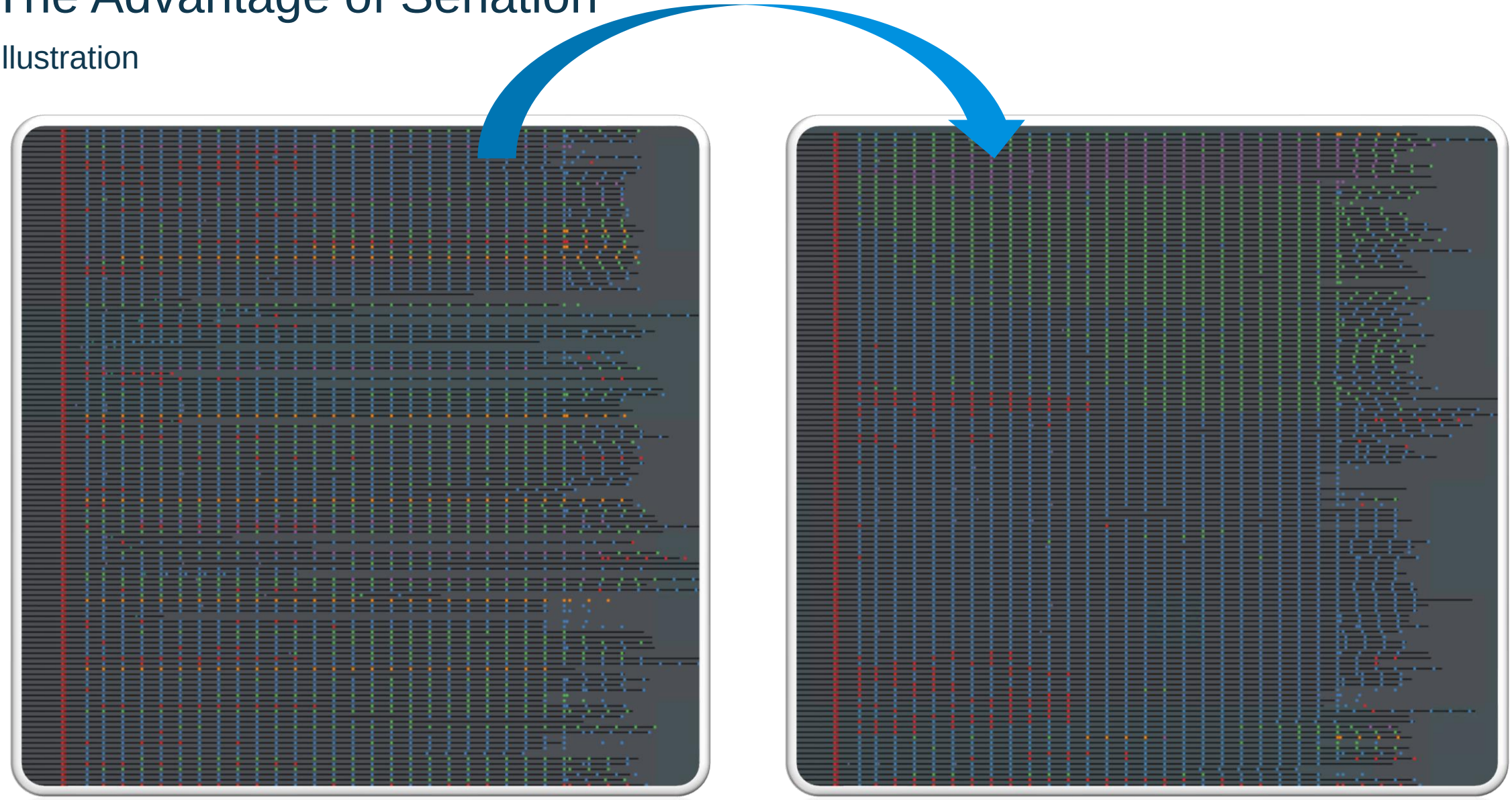
Why do we need Seriation?

Seriation in MEGAPLOTS

- The idea behind Megaplots:
  - Visualizing clinical data by showing individual patient data during the course of the study
  - Detection of patterns in the data, e.g. time-dependent effects
- Arranging subjects helps detecting patterns
- Two Steps:
  - Choose a distance measure to calculate distances between study courses of subjects
  - Choose seriation algorithm to locate similar study courses close to each other

The Advantage of Seriation

Illustration



Distance Measures

Different Types of Distance Measures

- The choice of a suitable distance measure is very important, since it is the foundation of the seriation
- Distances calculated with R package TraMineR
- Treating the study course of each subject as a state sequence

- There are three different methods to calculate the distance between two sequences:
  I. Distances between the distributions of the sequences
  II. Distances based on the count of common attributes
  III. Edit distances

Gabadinho, A., Ritschard, G., Müller, N. S., & Studer, M. (2011). Analyzing and Visualizing State Sequences in R with TraMineR. Journal of Statistical Software,40(4), 1-37.

Distances between the distributions of sequences

Distance Measures: Method I

- Let $p_{j|x}$ be the proportion of time spent in state j in sequence x
- p_j is the overall proportion of time spent in state j
- Manhattan Distance:
  - $d_{MAN}(x,y) = \sum_{j=1}^{|E|} |p_{j|x} - p_{j|y}|$
- X²-Distance:
  - $d_{CHI2}(x,y) = \sum_{j=1}^{|E|} \frac{(p_{j|x} - p_{j|y})^2}{p_j}$

Example:

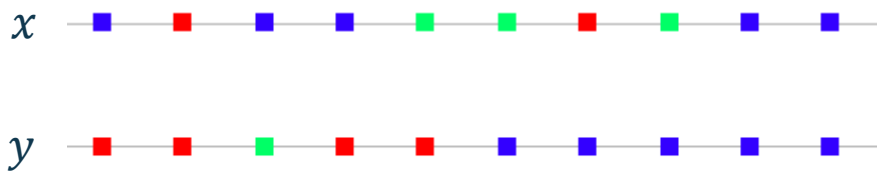
Distributions:
p_x = (0.2, 0.3, 0.5, 0)
p_y = (0.4, 0.1, 0.5, 0)
p = (0.3, 0.2, 0.5, 0)

Distances based on count of common attributes

Distance Measures: Method II

- Number of common attributes between sequence x and y: $A(x, y)$
- Distance: $d_A(x, y) = A(x, x) + A(y, y) - 2A(x, y)$
- Simple Hamming Distance
  - Only for Sequences of the same length
  - Counts the number of non-matching elements
- Longest Common Prefix
  - $A(x, y)$ is the number of successive common positions starting from the beginning of the sequences
- Longest Common Suffix
  - $A(x, y)$: number of common positions from the end
- Longest Common Subsequence
  - Number of Elements in one sequence that can be uniquely matched with elements occurring in the same order in the other sequence

Example:



Hamming Distance: $d_{HAM}(x, y) = 7$

Longest Common Prefix: $d_{LCP}(x, y) = 20$

Longest Common Suffix: $d_{RLCP}(x, y) = 16$

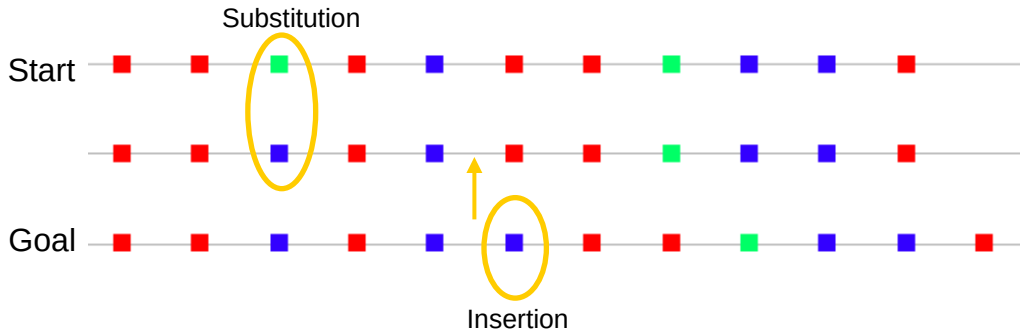
Longest Common Subsequence: $d_{LCS}(x, y) = 10$

Edit Distances - Overview

Distance Measures: Method III

Idea:

- Transform the one sequence into the other one
- For each transformation step, a cost is charged
- Distance: Minimal costs of transforming the one sequence into the other one



- Two types of possible operations/costs:
  - Substitution of state a by state b: $\gamma(a, b)$
  - Insertion or deletion (Indel) of state a: $c_{I(a)}$

Different types of edit distances:

- Optimal Matching $\mathbf{OM}$
  - Only substitution and indel possible
- Localized Optimal Matching $\mathbf{OMloc}$
- Spell-length-sensitive Optimal Matching $\mathbf{OMslen}$
- Optimal Matching of spell sequences $\mathbf{OMspell}$
- Optimal Matching of transition sequences $\mathbf{OMstran}$
- Dynamic Hamming Distance $\mathbf{DHD}$

Studer, M. & Ritschard, G. (2016). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures, Journal of the Royal Statistical Society, Series A, 179(2), 481-511. D

Edit Distances – Part 1

Distance Measures: Method III

Dynamic Hamming Distance DHD:

- No Insertion/Deletion → only for sequences of the same length
- Substitution cost depends on the position t in the sequence: calculate the cross-section of transition rates observed between position $t-1$ and t and between t and $t+1$
- Transition rates:
  - Higher cost for substituting between states when the transitions between them are rare; low cost when frequent transitions are observed
  - Transition rate between states a and b: probability $p(a|b)$ of switching from state a to state b between two successive positions

Localized Optimal Matching $OMloc$:

- Indel cost depends on the two adjacent states
  - Inserting/deleting a state similar to its neighbors would only change the length of the spell in that state → no effect on the sequencing
  - Insertion of a state different to its neighbors has more consequences → higher cost
  - Inserting state z between a and b:

$$c_I(z|a,b) = e\,\gamma_{max} + g\,\frac{\gamma(a,z)+\gamma(b,z)}{2}$$

  - γ_{max} maximal substitution cost
  - e cost of spell length transformation
  - g local insertion cost

Additional Parameters

Edit Distances – Part 2

Distance Measures: Method III

Spell-length-sensitive Optimal Matching **OMslen**:

- State a in a spell of length t will be denoted as a_t with a insertion/deletion cost defined as $\mathrm{c_I^H}(\mathrm{a}_t) = \frac{c_I(a_t)}{t^h}$
  - c_I basic insertion/deletion cost
  - h exponential weight of spell length
- Cost of substituting state a_{t_1} with b_{t_2}:

$$\gamma^H(a_{t_1}, b_{t_2}) = \gamma(a,b) \cdot link$$

  - $\gamma(a,b)$ basic substitution cost
  - $link$: function of $\frac{1}{\sqrt{t_1}}$ and $\frac{1}{\sqrt{t_2}}$
    - **mean** (default)
    - **gmean** (geometric mean)

Optimal Matching of spell sequences **OMspell**:

- Idea: consider for each different value of t a spell in state a during t units of time as a distinct element, denoted by a_t, of the alphabet
- Insertion/Deletion cost:

$$c_I^S(a_t) = c_I(a) + \delta \cdot (t-1)$$

- Substitution cost:

$$\gamma^S(a_{t_1}, b_{t_2}) = \begin{cases} \delta \cdot |t_1 - t_2| & if\ a = b \\ \gamma(a,b) + \delta \cdot (t_1 + t_2 - 2) & otherwise \end{cases}$$

  - $c_I(a)$ basic indel cost
  - $\gamma(a,b)$ basic substitution cost
  - δ cost of spell length transformation

Edit Distances – Part 3

Distance Measures: Method III

Optimal Matching of transition sequences **OMstran**:

- Idea: Instead of the states consider the transitions of the sequence
  - Example: Sequence: $a - a - b - b - c$
    Transitions: $aa - ab - bb - bc$
- Indel cost: $c_I^B(a \rightarrow b) = w\, c_I(a) + (1 - w)\, c_T(a \rightarrow b)$
- Substitution cost:

$$\gamma^B(a \rightarrow b, c \rightarrow d) = w\, \gamma(a, c) + (1 - w)\big(c_T(a \rightarrow b) + c_T(c \rightarrow d)\big)$$

- $c_I(a)$ indel cost of the origin state a
- $c_T(a \rightarrow b)$ transition type cost: **constant** (default), **subcost** (based on substitution cost), **prob** (based on transition probabilities)
- $\gamma(a, c)$ substitution cost between the origin states a and c
- $w \in [0,1]$ coefficient for controlling the trade-off between the cost related to the origin state and the cost related to the type of transition

Parameters for the Distance Measures – Part 1

Normalization

- d: distance between two sequences x and y
- $|x|, |y|$: length of the sequences x, resp. y
- m: maximal possible distance between two sequences of the length $|x|$ and $|y|$
- Possible values for the normalization parameter:
  - **none** (default)
  - **maxlength**: $\frac{d}{\max\{|x|,|y|\}}$
  - **gmean**: $1 - \frac{m-d}{\sqrt{|x| \cdot |y|}}$
  - **maxdist**: $\frac{d}{m}$
  - **YujianBo**: $2 \cdot \frac{d}{m+d}$
  - **auto**: **gmean** for distance measures **LCS, LCP, RCLP**, **maxlength** for distance measures **OM, HAM, DHD**, **YuhianBO** for distance measures **OMloc, OMslen, OMspell, OMstran**

Parameters for Distance Measures – Part 2

Insertion/Deletion Cost and Substitution Cost

- Insertion/Deletion Cost:
  - Possible values: **auto** (max(substitution cost) / 2), numeric value
- Substitution Cost:
  - **CONSTANT**: Same cost for all substitutions (default: 2)
  - **INDELS, INDELSLOG**: based on estimated indel cost
    - For each state a, derive state relative frequency f_a
    - **INDELS**: $indel_a = \frac{1}{f_a}$, **INDELSLOG**: $indel_a = \log\left[\frac{2}{1+f_a}\right]$
    - $\gamma(a,b) = indel_a + indel_b$
  - **TRATE**: derived from the observed transition rates
    - $P(a|b)$ probability of transition from state b to a
    - $\gamma(a,b) = 2 - P(a|b) - P(b|a)$
  - **ORDINAL:** states are considered ordinal, cost is proporional to distance of the two states

Overview of Distance Measures and their Parameters

Distance Measure	Parameters
OM	Substitution Cost, Insertion/Deletion Cost, Normalization
OMloc	Substitution Cost, Normalization, Cost of spell length transformation, Local Insertion Cost
OMslen	Substitution Cost, Insertion/Deletion Cost, Normalization, Substitution Cost Function, Exponential weight of spell length
OMspell	Substitution Cost, Insertion/Deletion Cost, Normalization, Cost of spell length transformation, Exponential weight of spell length
OMstran	Substitution Cost, Insertion/Deletion Cost, Normalization, Transition Indel Cost Method, Account for the transition from the previous state, Duplicate the last column, Origin-Transition Trade-Off weight
HAM, DHD	Substitution Cost, Normalization
CHI2	Normalization, Interval Length, Intervals Overlapping, Statedistribution as weights
EUCLID	Normalization, Interval Length, Intervals Overlapping
LCS, LCP, RLCP	Normalization

Seriation Algorithms

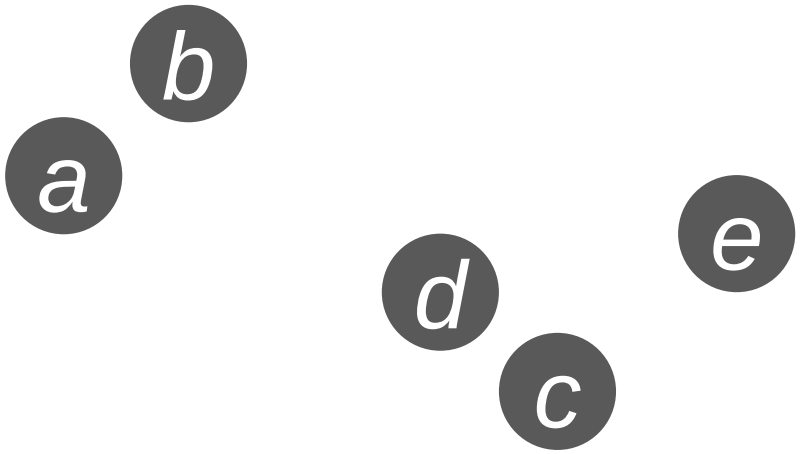
Seriation Algorithms

- Several seritation algorithms are available in Megaplots and can be selected by the user
- The implementation of these algorithms is based on the R package `seriation`
- Different Algorithms to arrange the subjects based on a matrix of pairwise distances:
  - Algorithms using hierarchical clustering and leaf ordering:
    - Hierarchical Clustering
    - Optimal-Leaf Ordering
    - Gruvaeus Wainer Heuristic
  - Algorithms minimizing the sum of distances
    - Visual Assessment of Tendency
    - Traveling Salesperson Problem

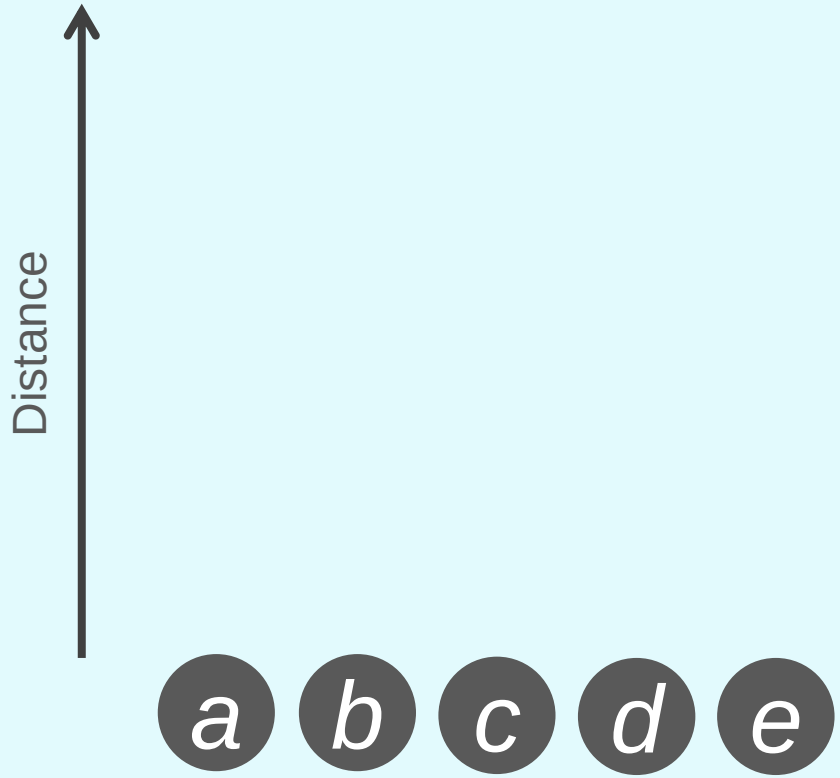
Hahsler M, Hornik K, Buchta C (2008). “Getting things in order: An introduction to the R package seriation.” Journal of Statistical Software, 25(3), 1–34. ISSN 1548-7660

Agglomerative Clustering

An introductory example

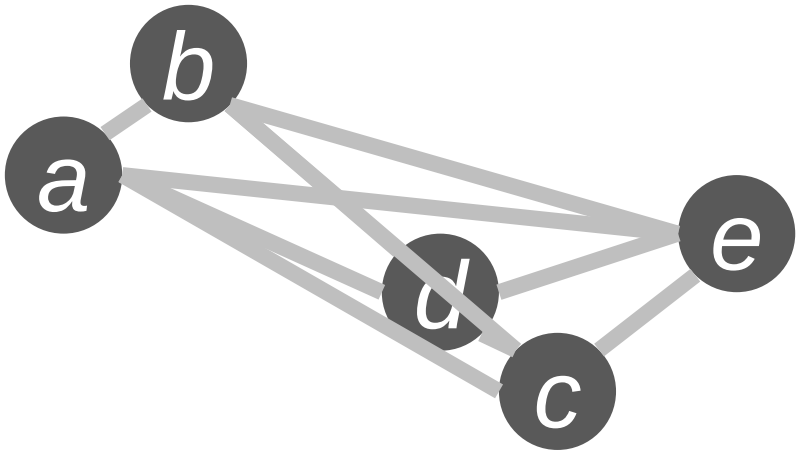


Each object forms an own cluster

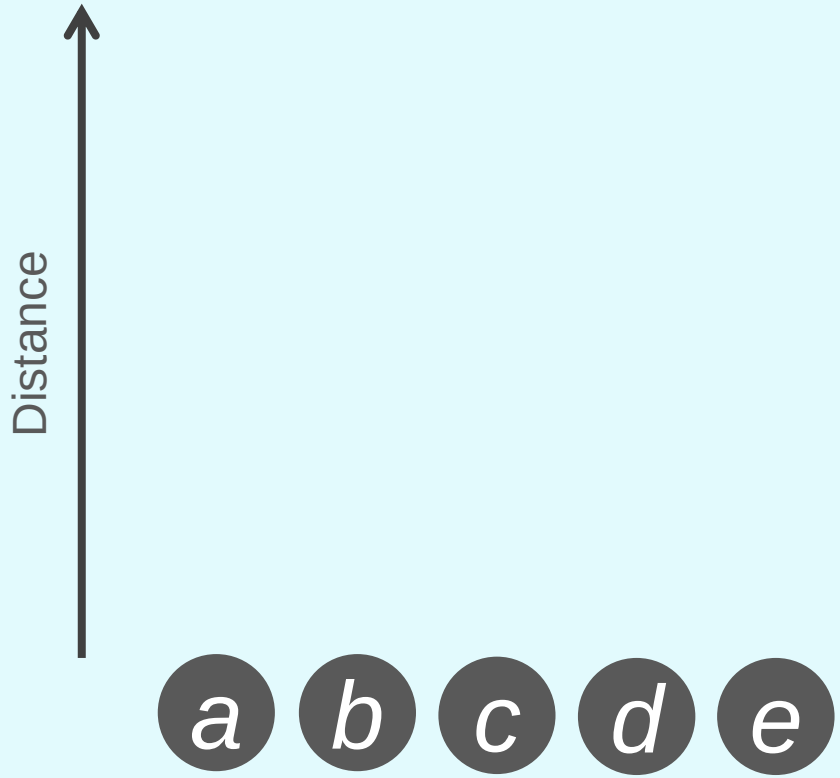


Agglomerative Clustering

An introductory example

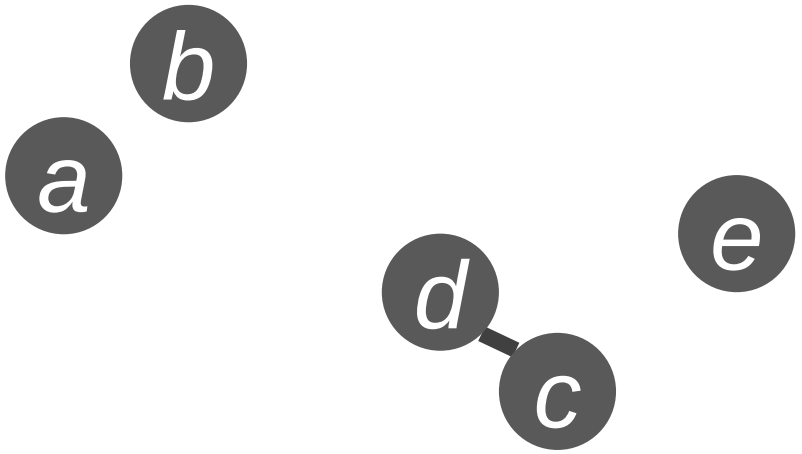


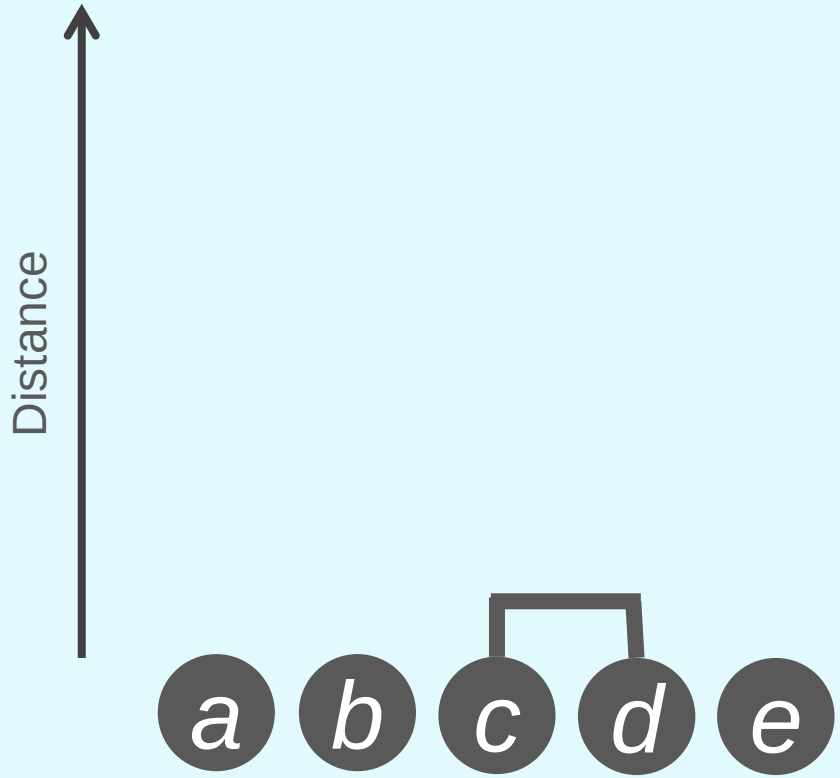
Compute all pairwise distances



Agglomerative Clustering

An introductory example

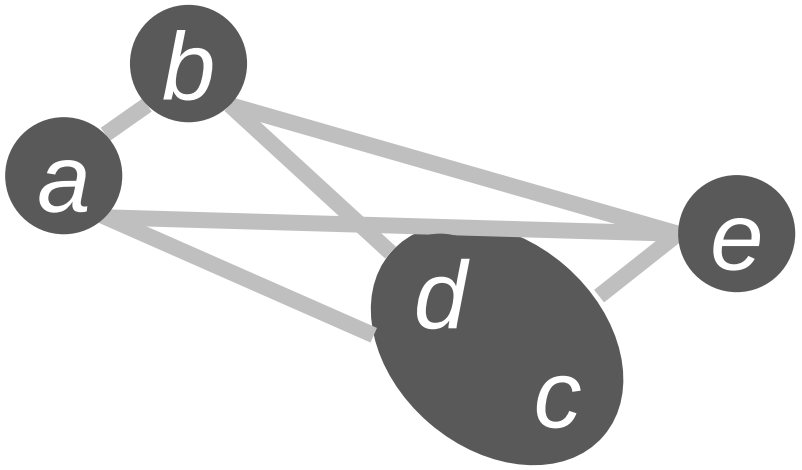




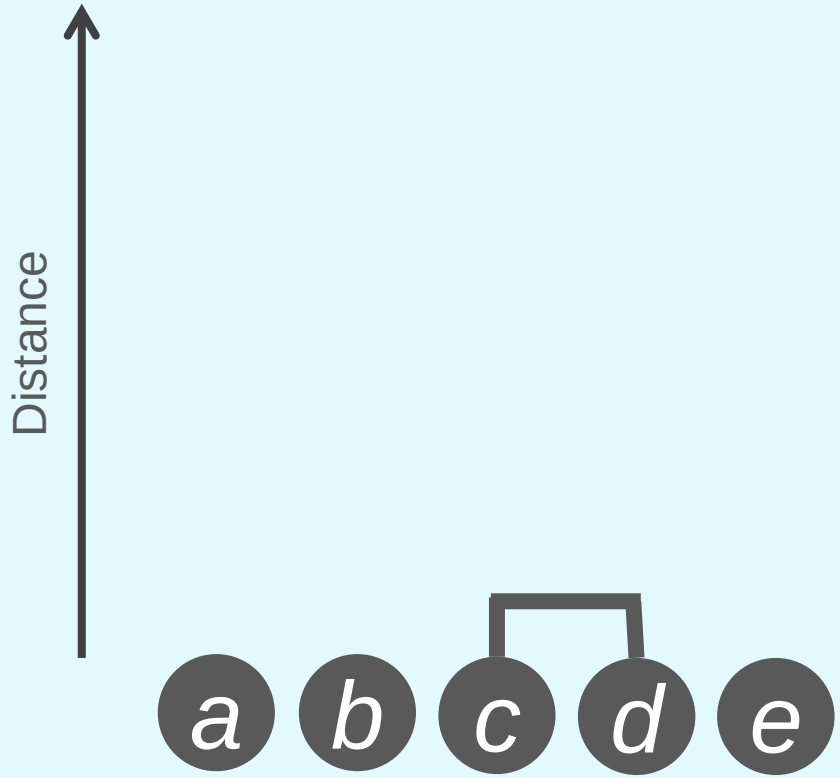
Join the two closest objects/clusters

Agglomerative Clustering

An introductory example

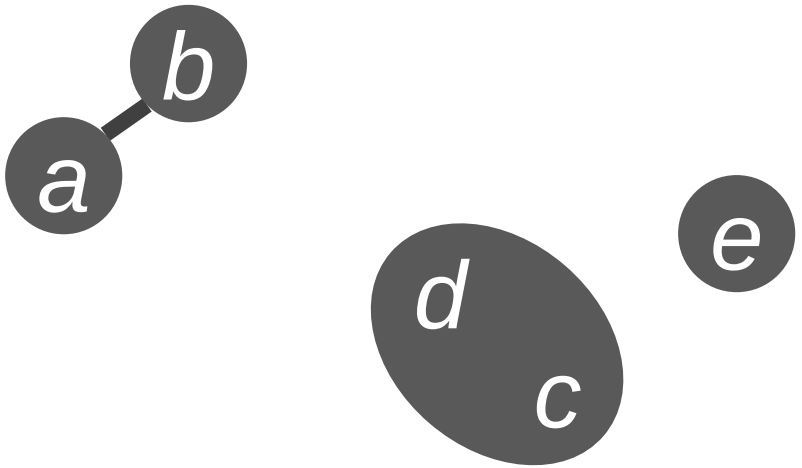


Compute all pairwise distances

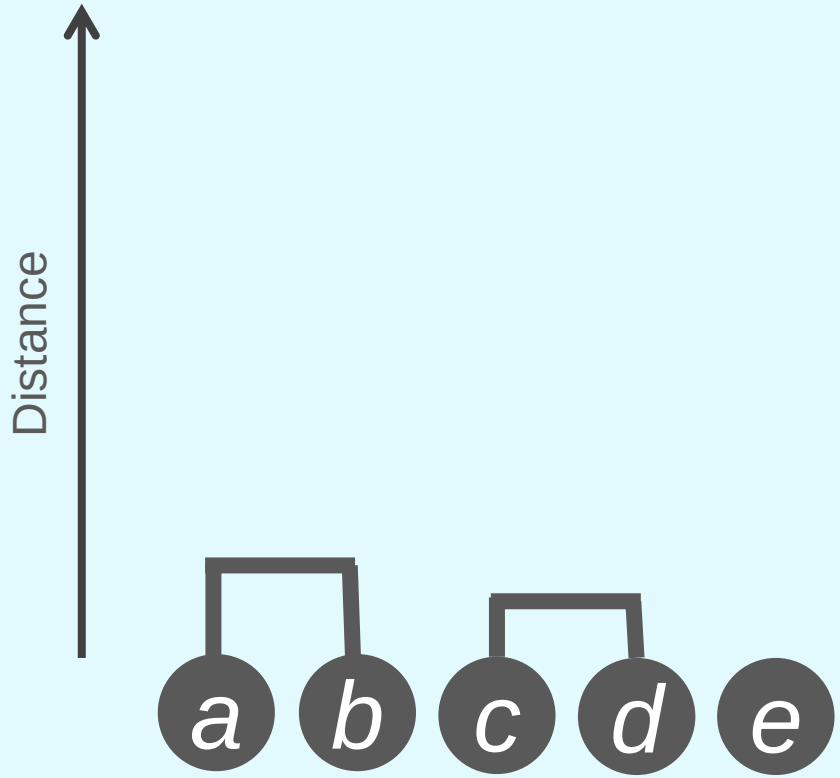


Agglomerative Clustering

An introductory example

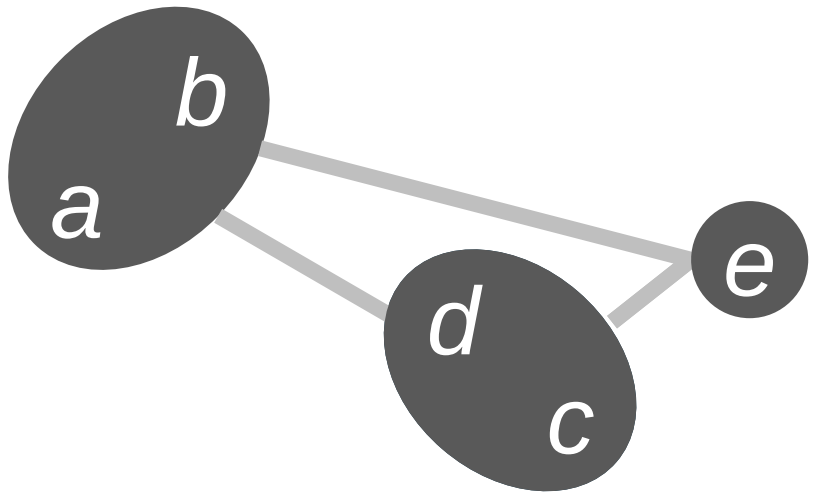


Join the two closest clusters

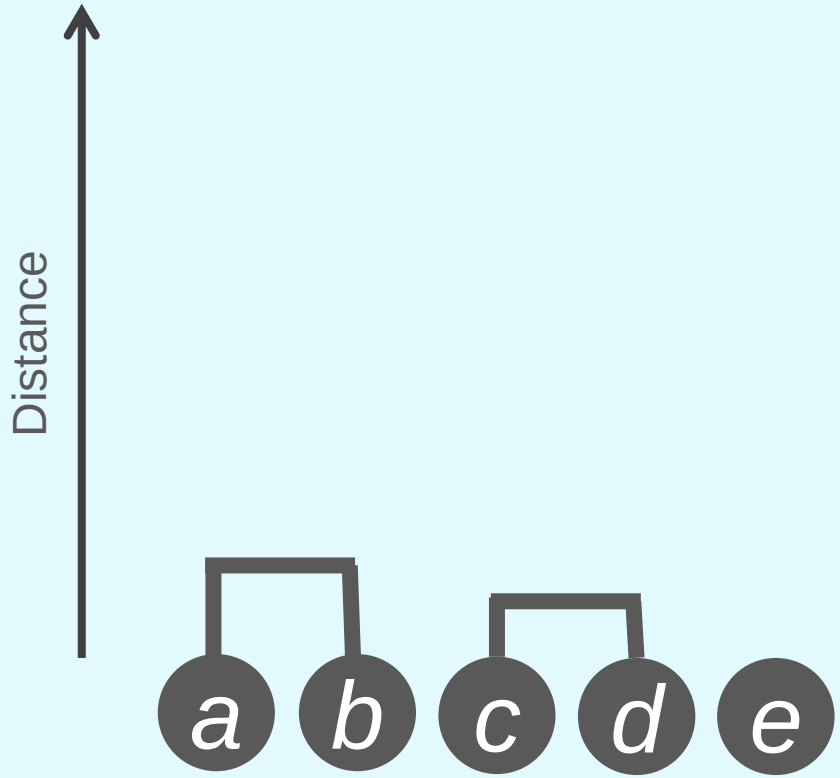


Agglomerative Clustering

An introductory example

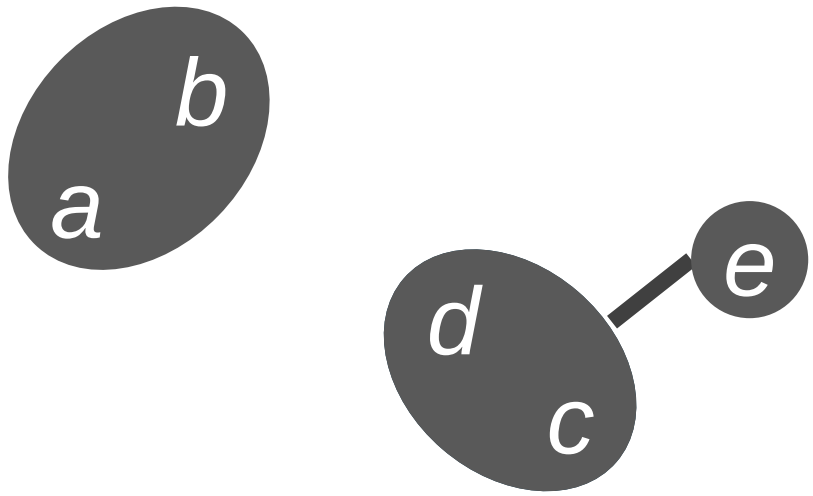


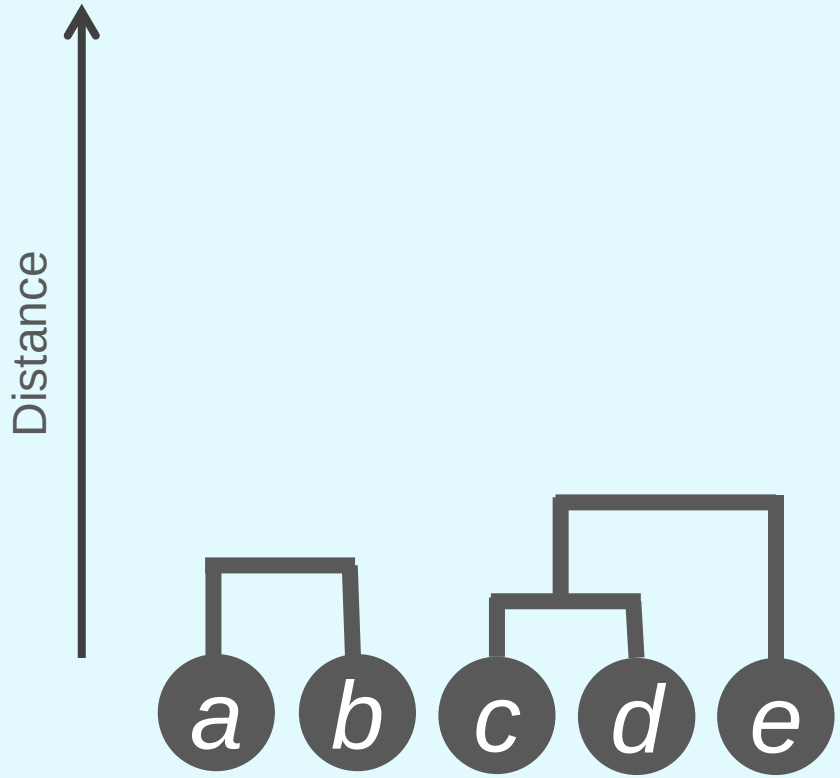
Compute all pairwise distances



Agglomerative Clustering

An introductory example

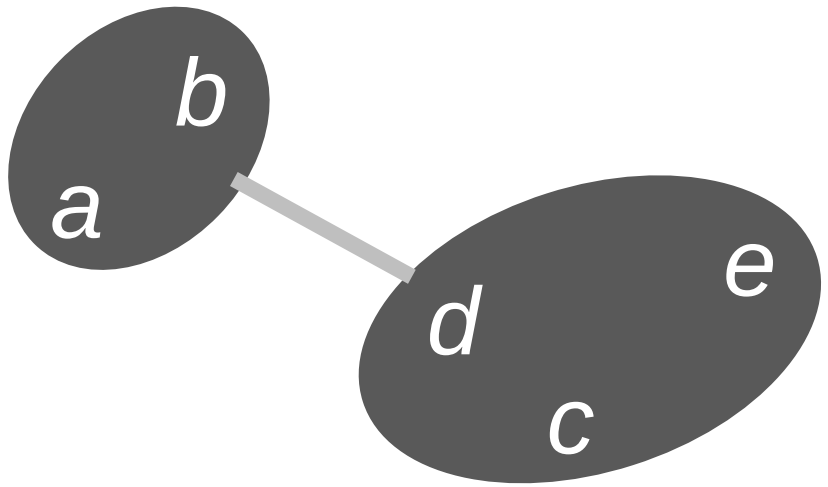




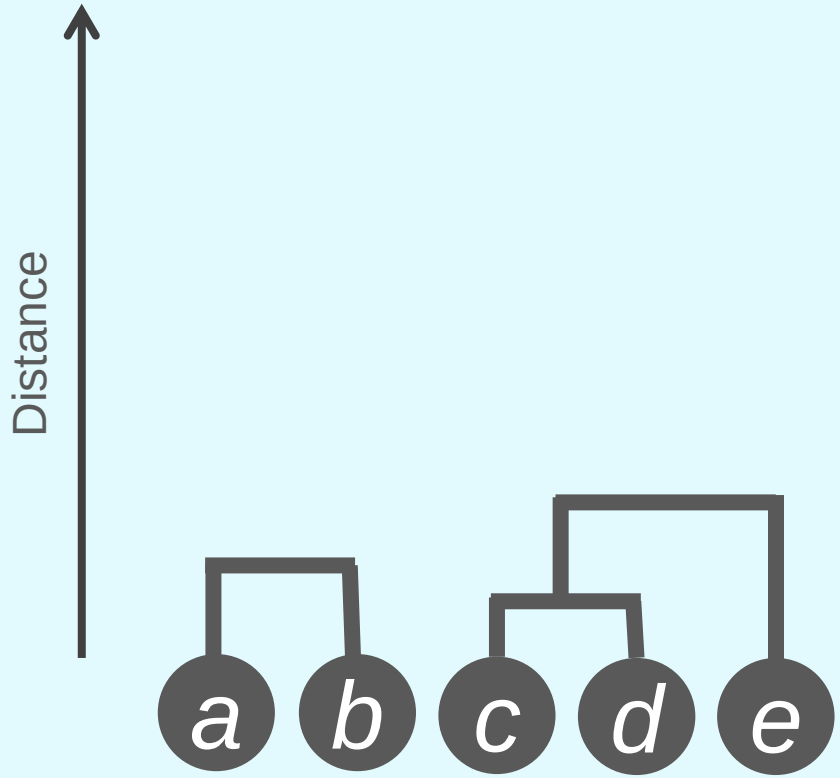
Join the two closest objects/clusters

Agglomerative Clustering

An introductory example

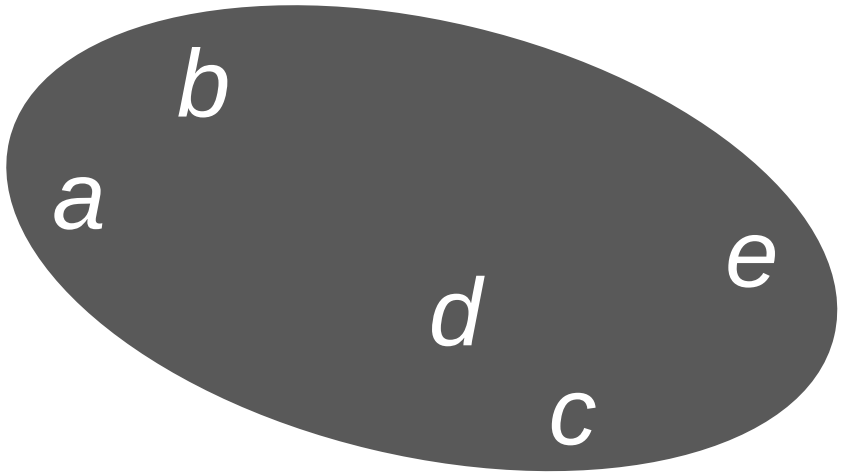


Compute all pairwise distances

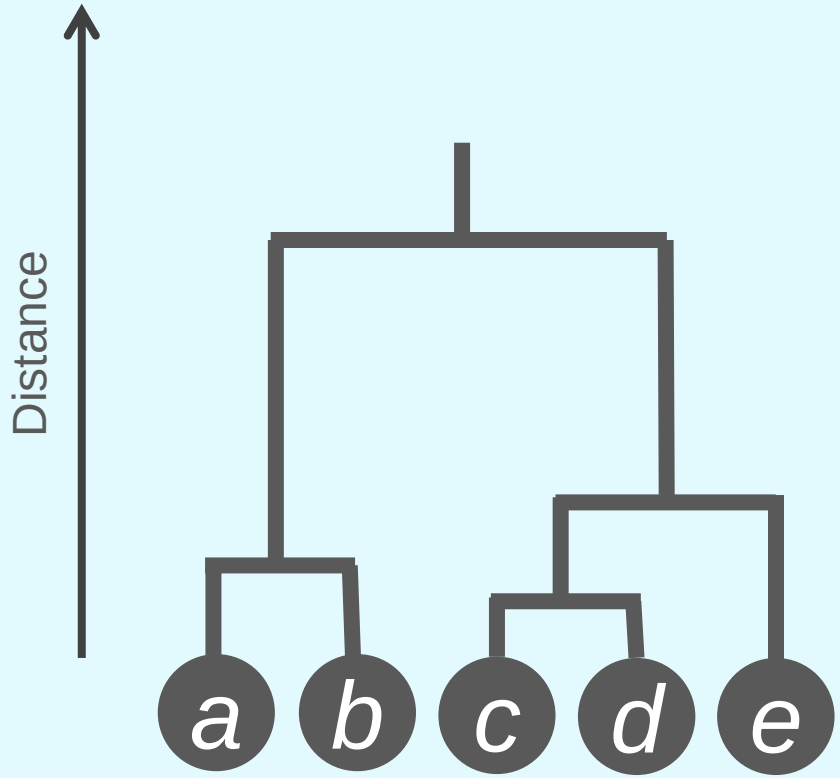


Agglomerative Clustering

An introductory example



Join the two closest objects/clusters



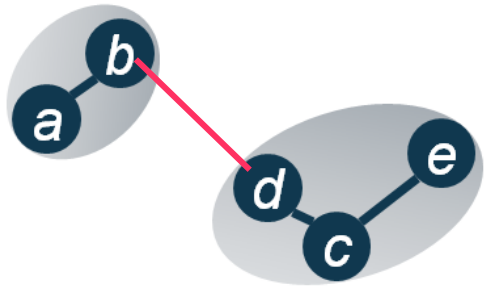
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$

Distances between two clusters

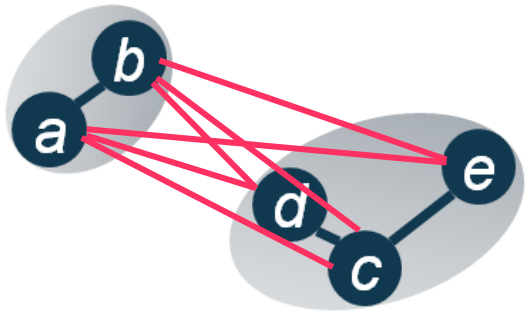
Linkage schemes

There are different methods to calculate the distance between two clusters containing multiple data points

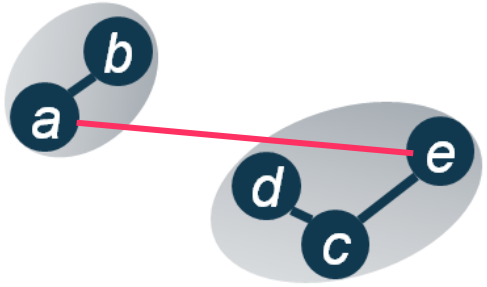
Single linkage



Average linkage



Complete linkage

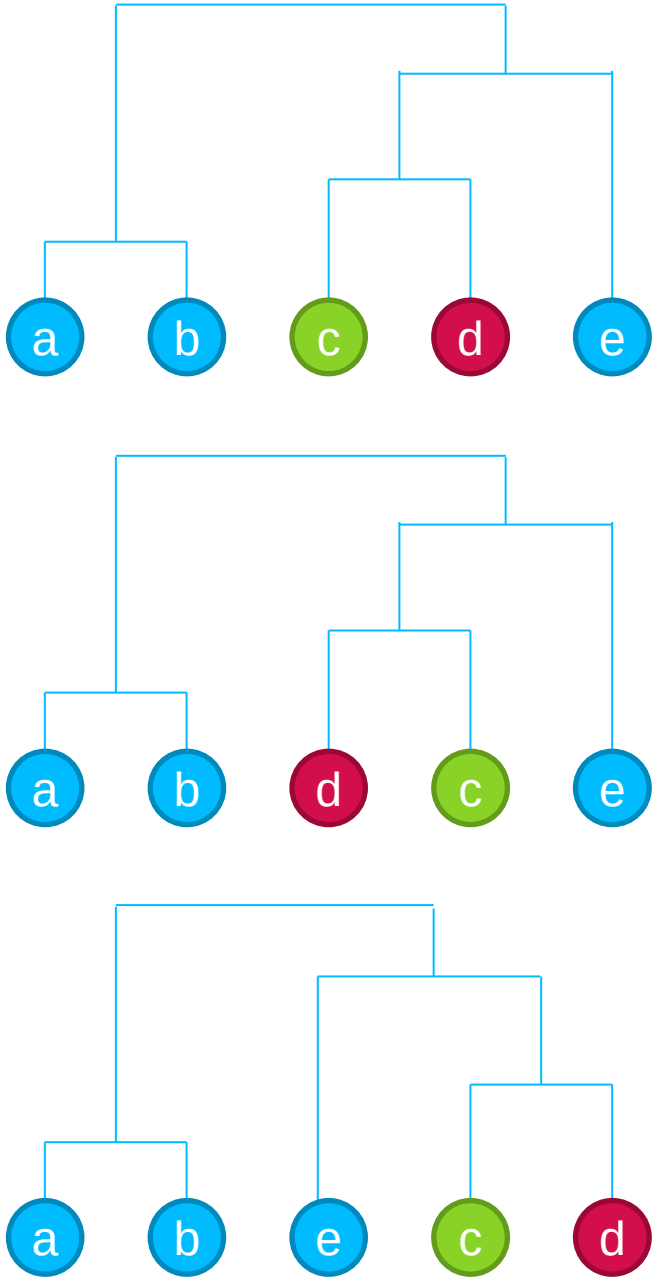


Ward linkage: $d(A,B) = \underbrace{\sum_{i \in A \cup B} d(x_i, m_{A \cup B})}_{\text{variance of the joint cluster A} \cup B} - \underbrace{\sum_{i \in A} d(x_i, m_A)}_{\text{variance of cluster } A} - \underbrace{\sum_{i \in B} d(x_i, m_B)}_{\text{variance of cluster } B}$

d : distance measure
m_j: center of cluster j

Dendrogram

- A dendrogram is a graphical tool to display the results of hierarchical clustering
- The order of leafs can be used for sorting the subjects in the Megaplots display
- BUT: a dendrogram is not unique!
  There are 2^{n-1} possible orderings consistent with the tree structure
- Rotation methods are used to sort leafs such that neighbored leafs are most similar



Rotation Methods for Dendrograms

Optimal Leaf Ordering **OLO**:

- Idea: Maximize the sum of the similarity of adjacent elements
- Problem definition:
  - Tree T with $n-1$ internal leafs nodes, that can be flipped
  - Φ space of 2^{n-1} possible orderings
  - For $\phi \in \Phi$ define $D^{\phi}(T) = \sum_{i=1}^{n-1} S(z_{\phi_i}, z_{\phi_{i+1}})$
  - z_{ϕ_i} is the i^{th} leaf when T is ordered according to ϕ and S measures the similarity
  - → Maximize $D^{\phi}(T)$

Gravaeus Wainer Heuristic **GW**:

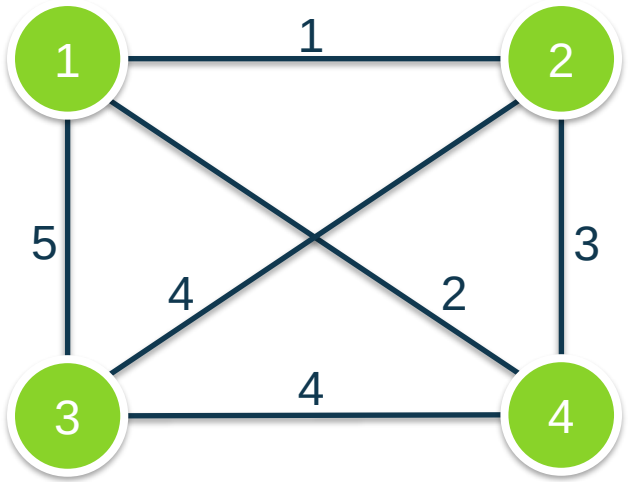
- Idea: Order the branches of the dendrogram in such way that the subjects at the edge of adjacent subtrees are most similar
- Minimizes hamiltonian path length
- Produces an unique order
- Faster than OLO but less optimal

Distance Matrix as a Graph

- With a chosen distance measure, we get pairwise distances between all subjects
- Let d_{ij} be the distance between subject i and subject j
- Distance matrix $D = (d_{ij})_{i,j=1,\dots,n}$

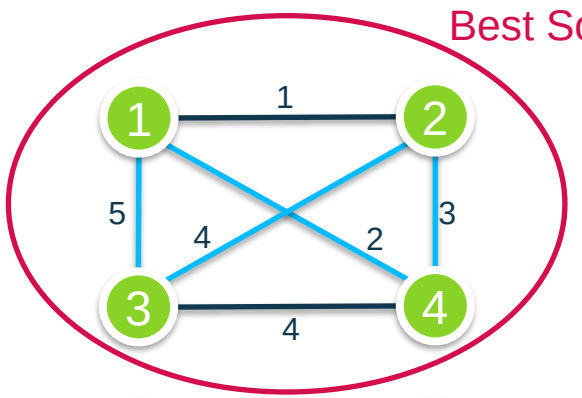
$$\mathrm{D} = \begin{pmatrix} 0 & 1 & 5 & 2 \\ 1 & 0 & 4 & 3 \\ 5 & 4 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{pmatrix}$$

- Display distance matrix as undirected graph with subjects as vertices and distances as weights at the edges between the two vertices

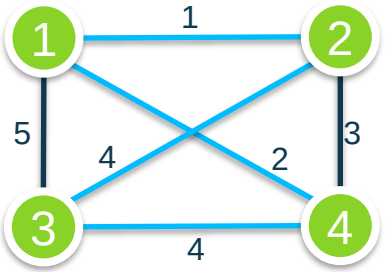


Traveling Salesperson Problem TSP

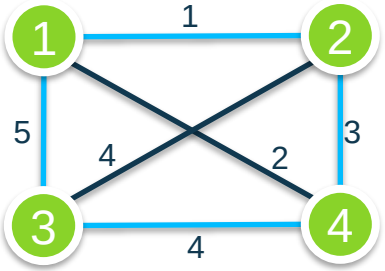
- Visit all subject-vertices once and get back to start subject
- For each edge traveled, sum up the weights → Minimize cost C
- Find the optimal route with the 2-opt technique:
  - Start with a random tour and iteratively swap out two edges represented by the distance matrix till no more improvements are possible
  - Repeat 10 times and choose best solution
- Since the output is a connected circle, find best cutting point: add dummy subject with equal distance to all other subjects and cut where the dummy city is located
- Arrange subjects in the order they where visited



C = 11



C = 14



C = 13

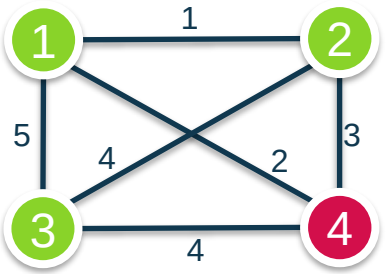
Hahsler M, Hornik K (2022). TSP: Traveling Salesperson Problem (TSP). R package version 1.2-1

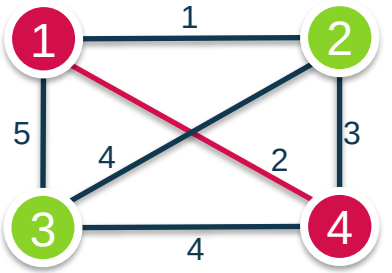
Hahsler M, Hornik K (2007). "TSP - Infrastructure for the traveling salesperson problem." Journal of Statistical Software, *23*(2), 1-21. ISSN 1548-7660

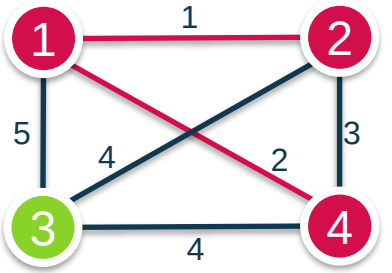
Climer, Sharlee & Zhang, Weixiong. (2006). Rearrangement Clustering: Pitfalls, Remedies, and Applications. Journal of Machine Learning Research. 7. 919-943.

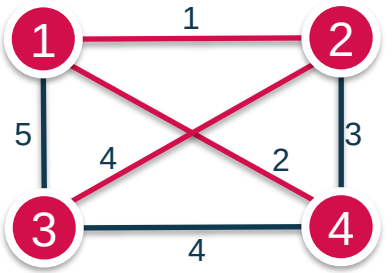
Visual Assessment of Tendency VAT

- Minimum Spanning Tree: Connects all vertices together, without any cycles and with the minimum possible total edge weight
- Prims‘s Algorithm:
  - Choose an arbitrary vertex
  - Grow the tree by one edge: find the edge that connects a new vertex to the tree with minimal edge weight and add it to the tree
  - Repeat until all vertices are part of the tree
- The order of the subjects is given by the order the subjects are added to the minimum spanning tree







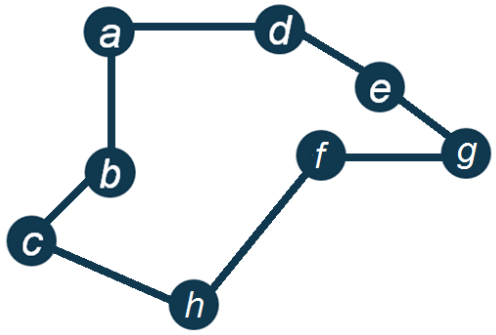


Bezdek, James & Hathaway, R.J.. (2002). VAT: A tool for visual assessment of (cluster) tendency. Proceedings of the International Joint Conference on Neural Networks. 3. 2225 - 2230.

Anti-Robinson seriation by simulated annealing ARSA

- Idea:
  - For any possible seriation count the number of violations
  - Find the smallest number of violations from this concept → Choose corresponding seriation
  - In practice there are many possible seriations such that not all can be evaluated
  - → Use a partial enumeration method simulated annealing

- Number of violations
  - Get a seriation for all subjects
  - Check for each subject, if the order of the distances to the other subjects corresponds to the distance in the seriation
  - Example:
    - Seriation: $h - c - b - a - d - e - g - f$
    - e and f are more similar than f and g, but in the seriation g is between e and f
    - d and f are more similar than d and a



Brusco M, Köhn HF, Stahl S (2008). “Heuristic Implementation of Dynamic Programming for Matrix Permutation Problems in Combinatorial Data Analysis.” Psychometrika, 73(3), 503–522.

References

// Bezdek, James & Hathaway, R.J.. (2002). VAT: A tool for visual assessment of (cluster) tendency. Proceedings of the International Joint Conference on Neural Networks. 3. 2225 - 2230.

// Brusco M, Köhn HF, Stahl S (2008). “Heuristic Implementation of Dynamic Programming for Matrix Permutation Problems in Combinatorial Data Analysis.” Psychometrika, 73(3), 503–522.

// Climer, Sharlee & Zhang, Weixiong. (2006). Rearrangement Clustering: Pitfalls, Remedies, and Applications. Journal of Machine Learning Research. 7. 919-943.

// Gabadinho, A., Ritschard, G., Müller, N. S., & Studer, M. (2011). Analyzing and Visualizing State Sequences in R with TraMineR. Journal of Statistical Software, 40(4), 1-37

// Hahsler M, Hornik K, Buchta C (2008). “Getting things in order: An introduction to the R package seriation.” Journal of Statistical Software, 25(3), 1–34. ISSN 1548-7660

// Hahsler M, Hornik K (2022). TSP: Traveling Salesperson Problem (TSP). R package version 1.2-1

// Hahsler M, Hornik K (2007). “TSP - Infrastructure for the traveling salesperson problem.” Journal of Statistical Software, *23*(2), 1-21. ISSN 1548-7660

// Studer, M. & Ritschard, G. (2016). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures, Journal of the Royal Statistical Society, Series A, 179(2), 481-511.

// Ziv Bar-Joseph, David K. Gifford, Tommi S. Jaakkola, Fast optimal leaf ordering for hierarchical clustering , Bioinformatics, Volume 17, Issue suppl_1, June 2001, Pages S22–S29