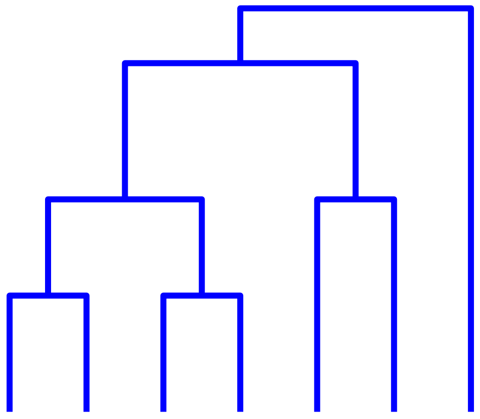
Announcements

- Recall: if you are registered *or* auditing (on at least a semiregular basis), email me from you preferred email address.
- Check the class web page for a more detailed schedule, optional readings, class slides.
- For next time: Read Eisen *et al.* (1998) “Cluster analysis and display of genome-wide expression patterns.” Proceedings of the National Academy of Sciences of the USA, Vol. 95, pp. 14863–14868, and write a 1-2 page critique.
- Today: Hierarchical clustering, SOMs, Multi-dimensional scaling.

Hierarchical clustering

- Organizes data objects into trees.
- For visualization, exploratory data analysis.
- “Agglomerative” methods build the tree bottom-up, successively grouping together clustering deemed most similar.
- “Divisive” methods build the tree top-down, recursively partitioning the data.



What is a hierarchical clustering?

- Given data objects $D = \{x_1, x_2, \ldots, x_n\}$.
- A hierarchical clustering is a set of subsets (clusters) of D, $C = \{C_1, C_2, \ldots, C_m\}$, where
  - $D \in C$
  - The C_j can be assigned to the nodes of a tree such that the cluster at any node is precisely the union of the clusters at the node's children (if any).

Example of a hierarchical clustering

- Suppose $D = \{1, 2, 3, 4, 5, 6, 7\}$.
- One hierarchical clustering is $C = \{\{1\}, \{2, 3\}, \{4, 5\}, \{1, 2, 3, 4, 5\}, \{6, 7\}, \{1, 2, 3, 4, 5, 6, 7\}\}$.
- Leaves of the tree need not correspond to single data objects.
- The branching factor of the tree is not limited.

$\Rightarrow$ However, most hierarchical clustering algorithms produce binary trees, and take single data objects as the smallest clusters.

Agglomerative clustering

- Inputs: A set of data objects, and pairwise distances $d(x, x')$ between them.
- Outputs: A hierarchical clustering
- Algorithm:
  - Begin by putting each object as its own cluster on a working list W.
  - Repeat
    - ∗ Find the two clusters in W that are most “similar”.
    - ∗ Remove them from W.
    - ∗ Add their union to W.

    Until W contains a single cluster with all the data objects.
  - The hierarchical clustering comprises all clusters appearing in W at any stage of the algorithm.

How do we measure similarity between clusters?

Let $C_1 = \{x_1, x_2, \dots, x_m\}$ and $C_2 = \{x'_1, x'_2, \dots, x'_n\}$.
Three common measures of the *dissimilarity* are:

- Distance between nearest objects (“Single-linkage” agglomerative clustering, or “nearest neighbor”):

$$\min_{x \in C_1, x' \in C_2} d(x, x')$$

- Distance between farthest objects (“Complete-linkage” agglomerative clustering, or “furthest neighbor”):

$$\max_{x \in C_1, x' \in C_2} d(x, x')$$

- Average distance between objects (“Group-average” agglomerative clustering):

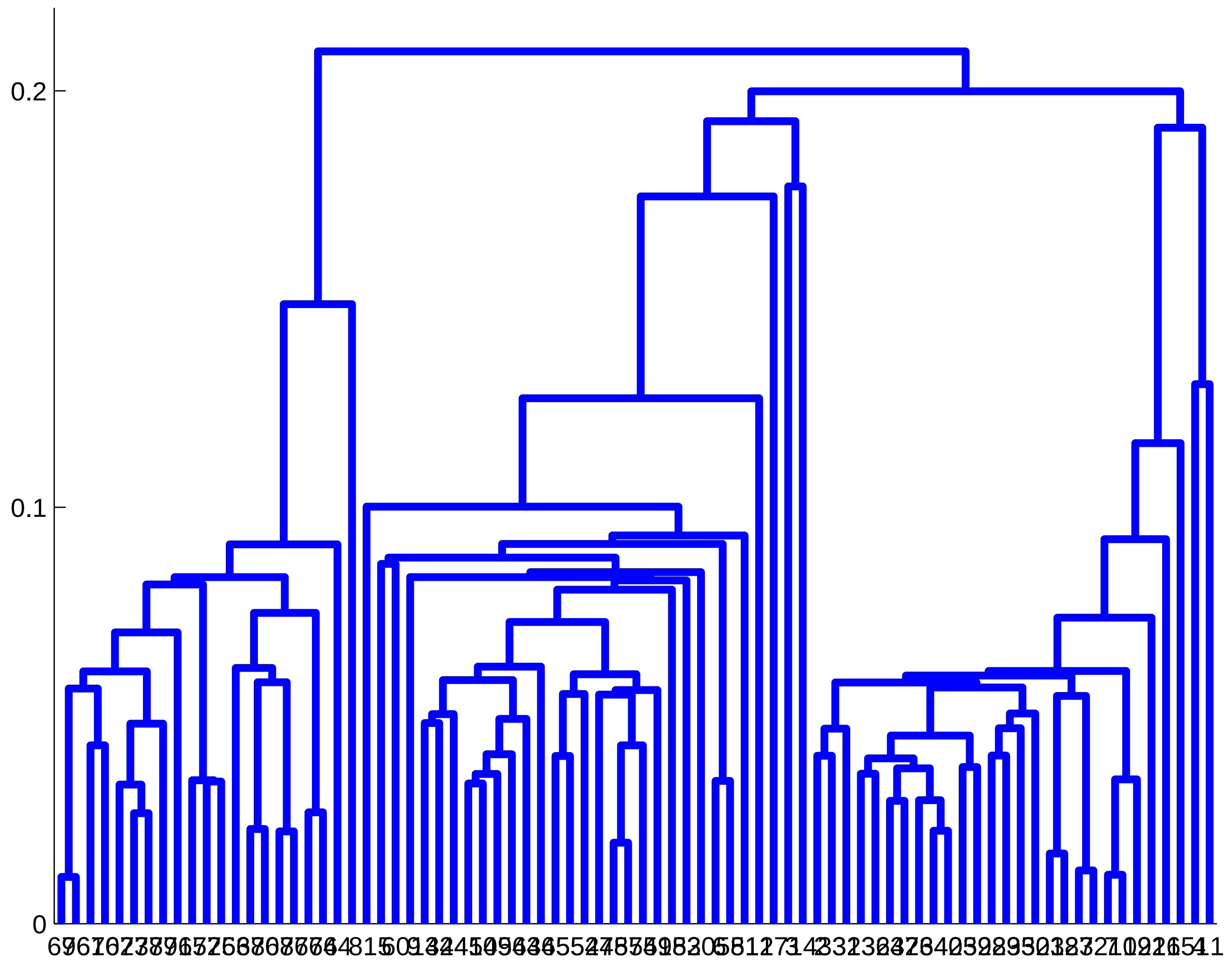
$$\frac{1}{mn} \sum_{x \in C_1, x' \in C_2} d(x, x')$$

[Show examples!]

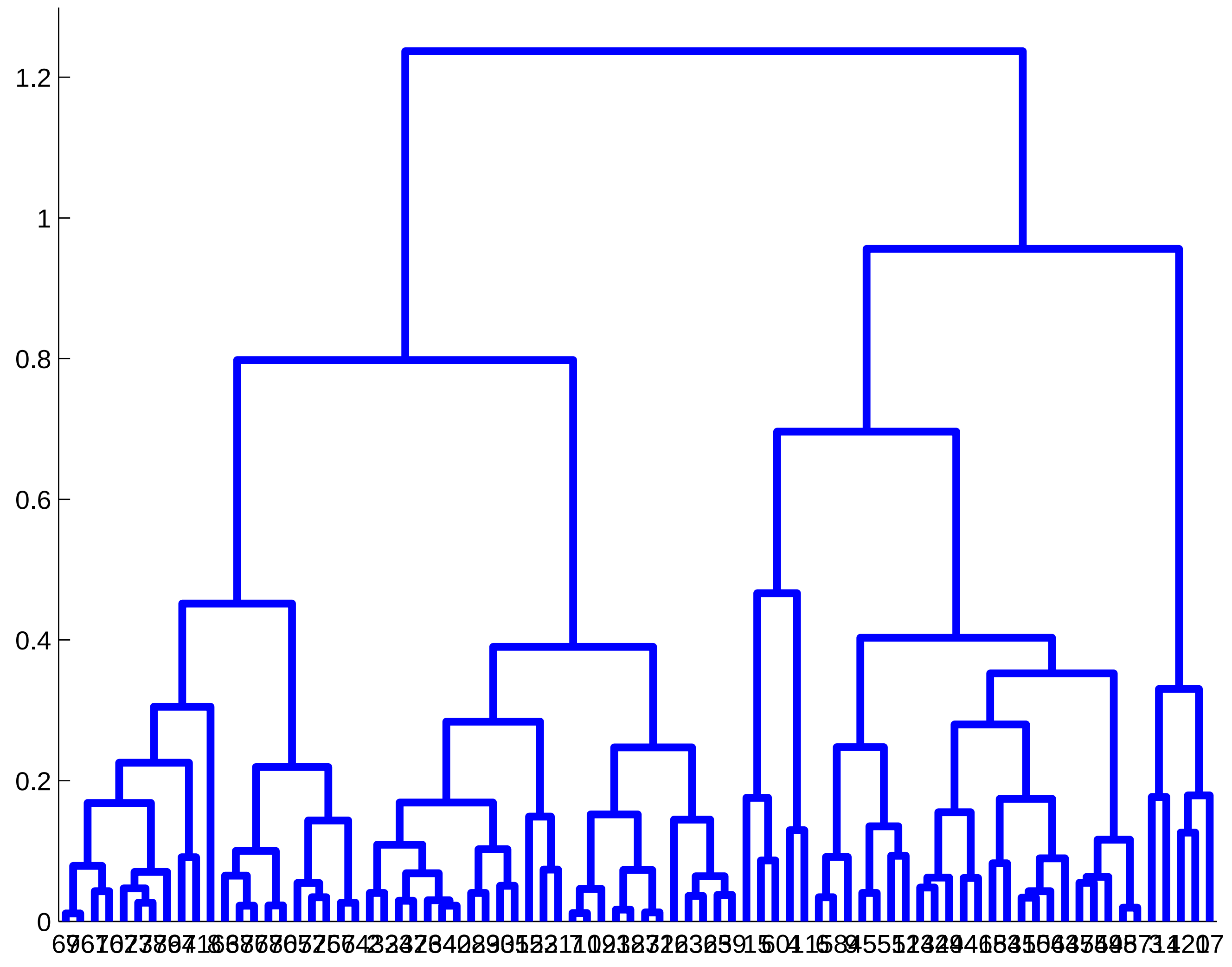
Dendrograms and Monotonicity

- Single-linkage, complete-linkage and group-average dissimilarity measure all share a monotonicity property:
  - Let A, B, C be clusters.
  - Let $\mathcal{DS}$ be one of the dissimilarity measures.
  - If $\mathcal{DS}(A, B) < \mathcal{DS}(A, C)$ and $\mathcal{DS}(A, B) < \mathcal{DS}(B, C)$, then $\mathcal{DS}(A, B) < \mathcal{DS}(A \cup B, C)$.
- Implication: every time agglomerative clustering merges two clusters, the dissimilarity of those clusters is $\geq$ the dissimilarity of all previous merges.
- Dendrograms (trees depicting hierarchical clusterings) are often drawn so that the height of a node corresponds to the dissimilarity of the merged clusters.

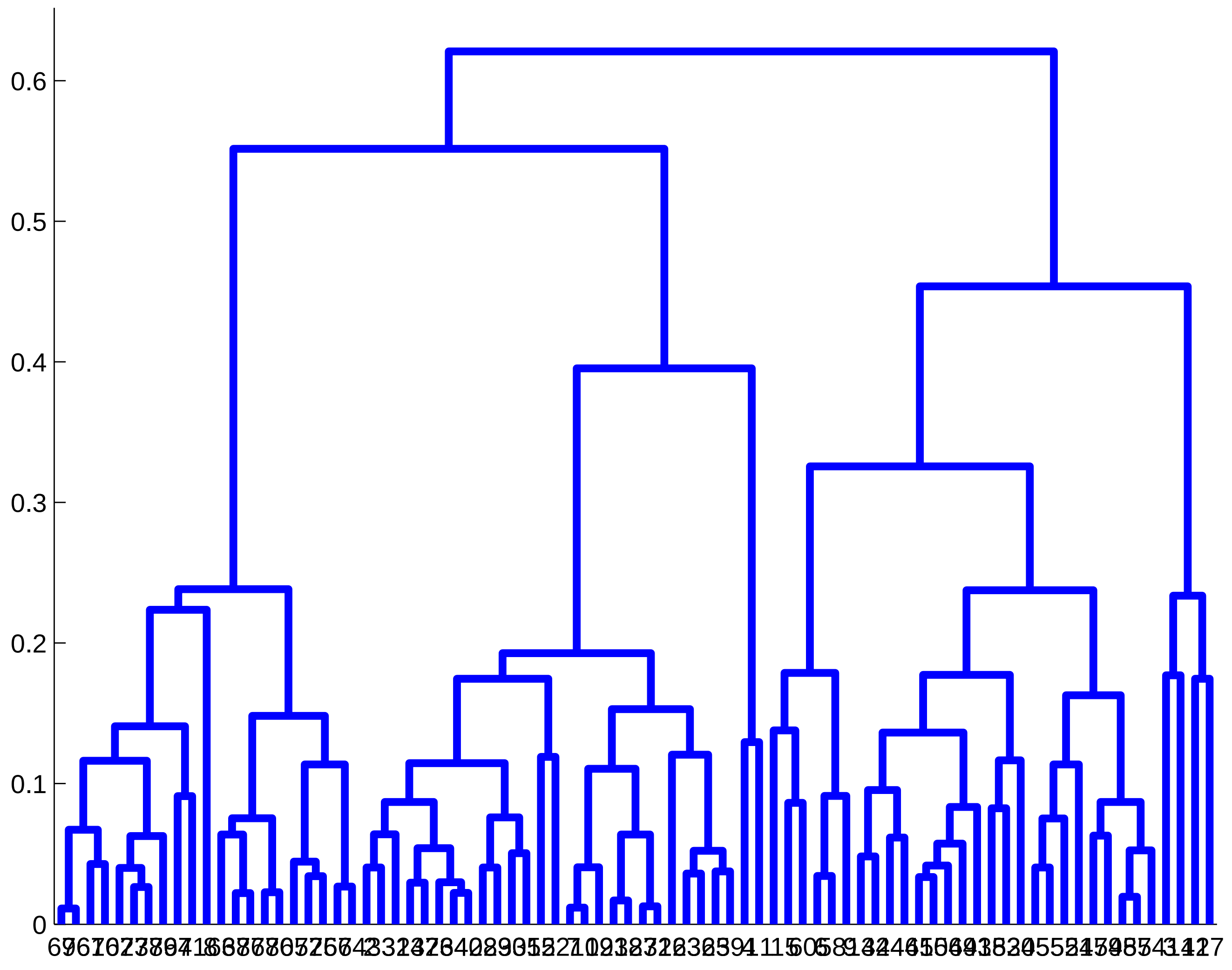
Dendrogram for single-linkage clustering of Example 1



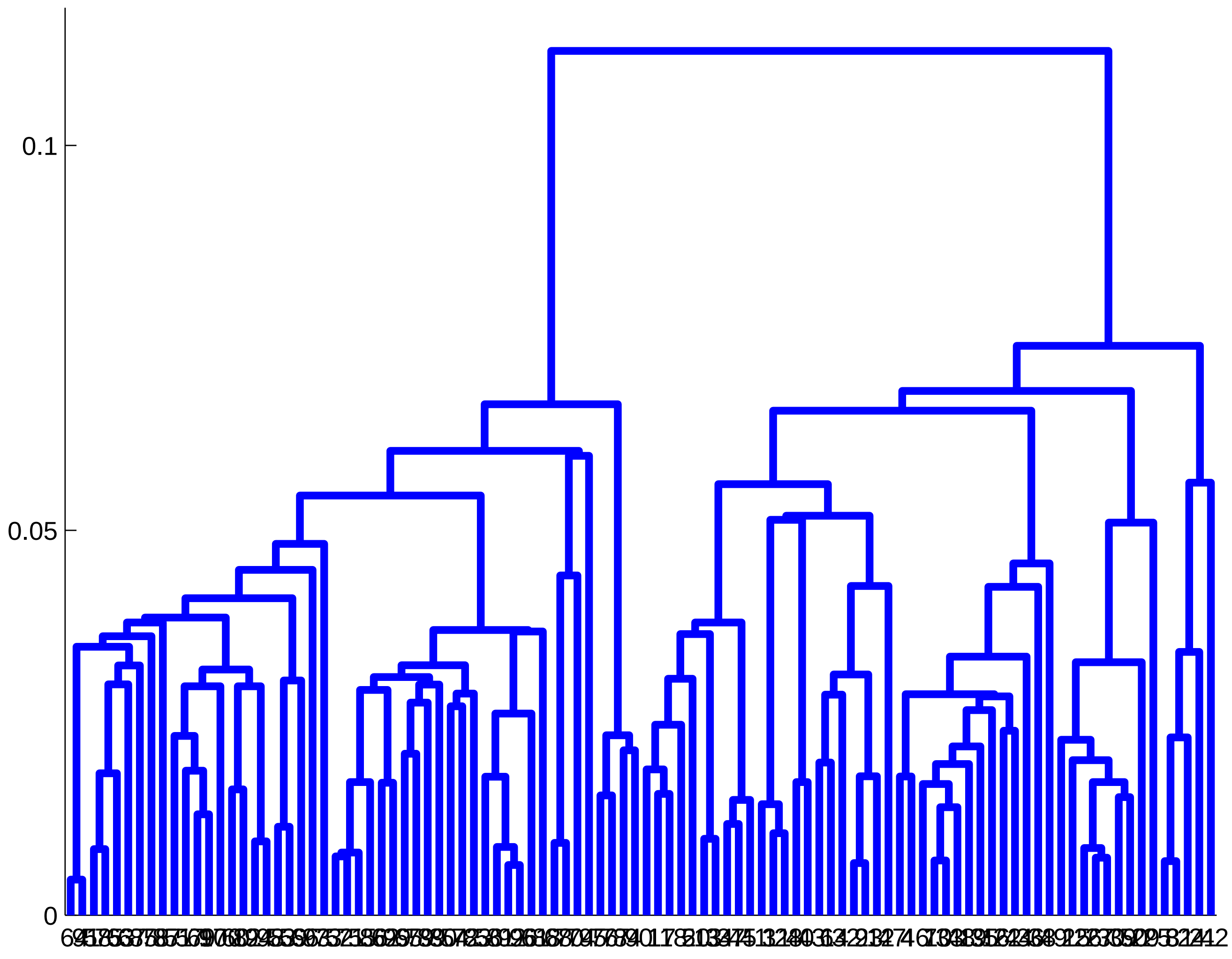
Dendrogram for complete-linkage clustering of Example 1



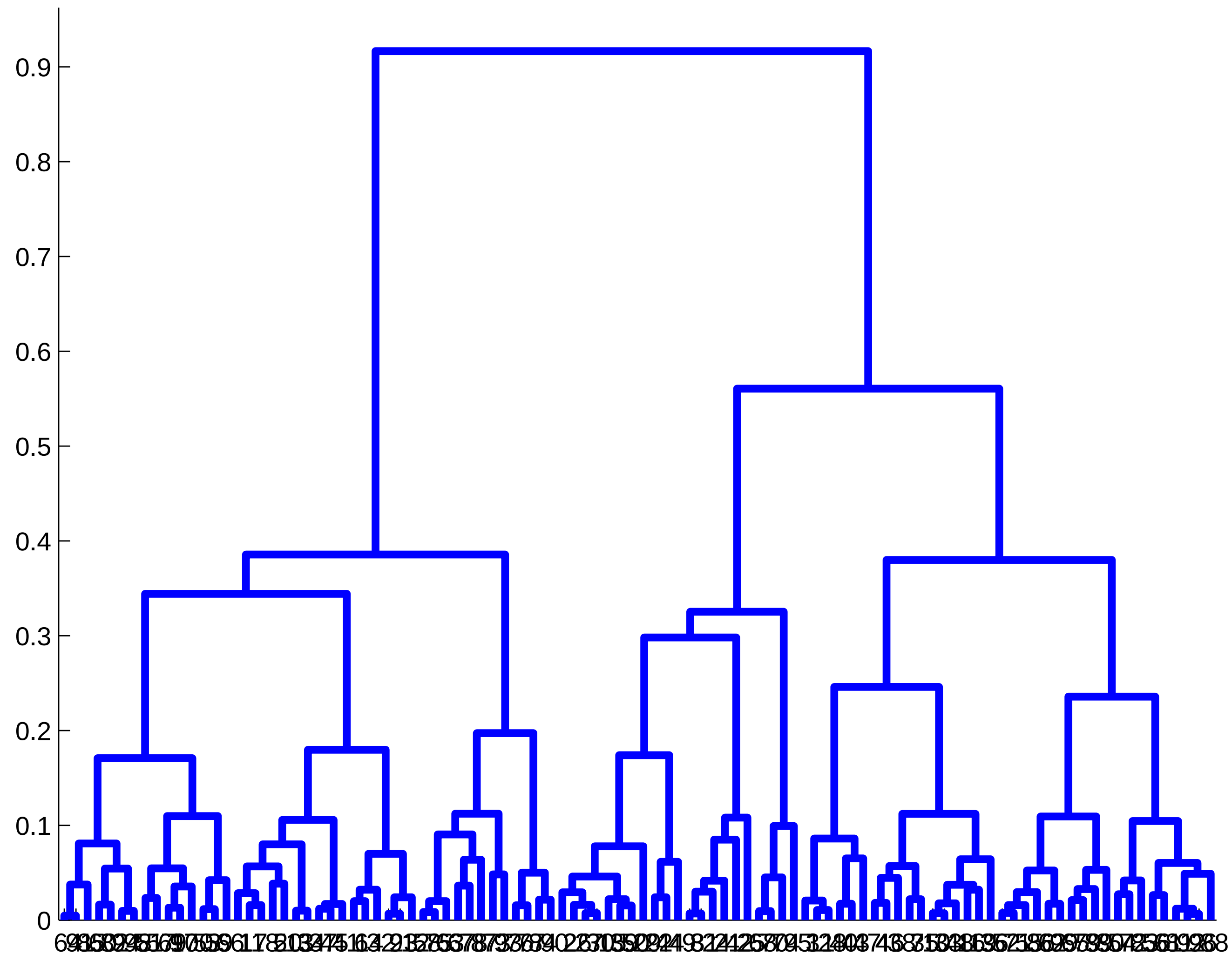
Dendrogram for average-linkage clustering of Example 1



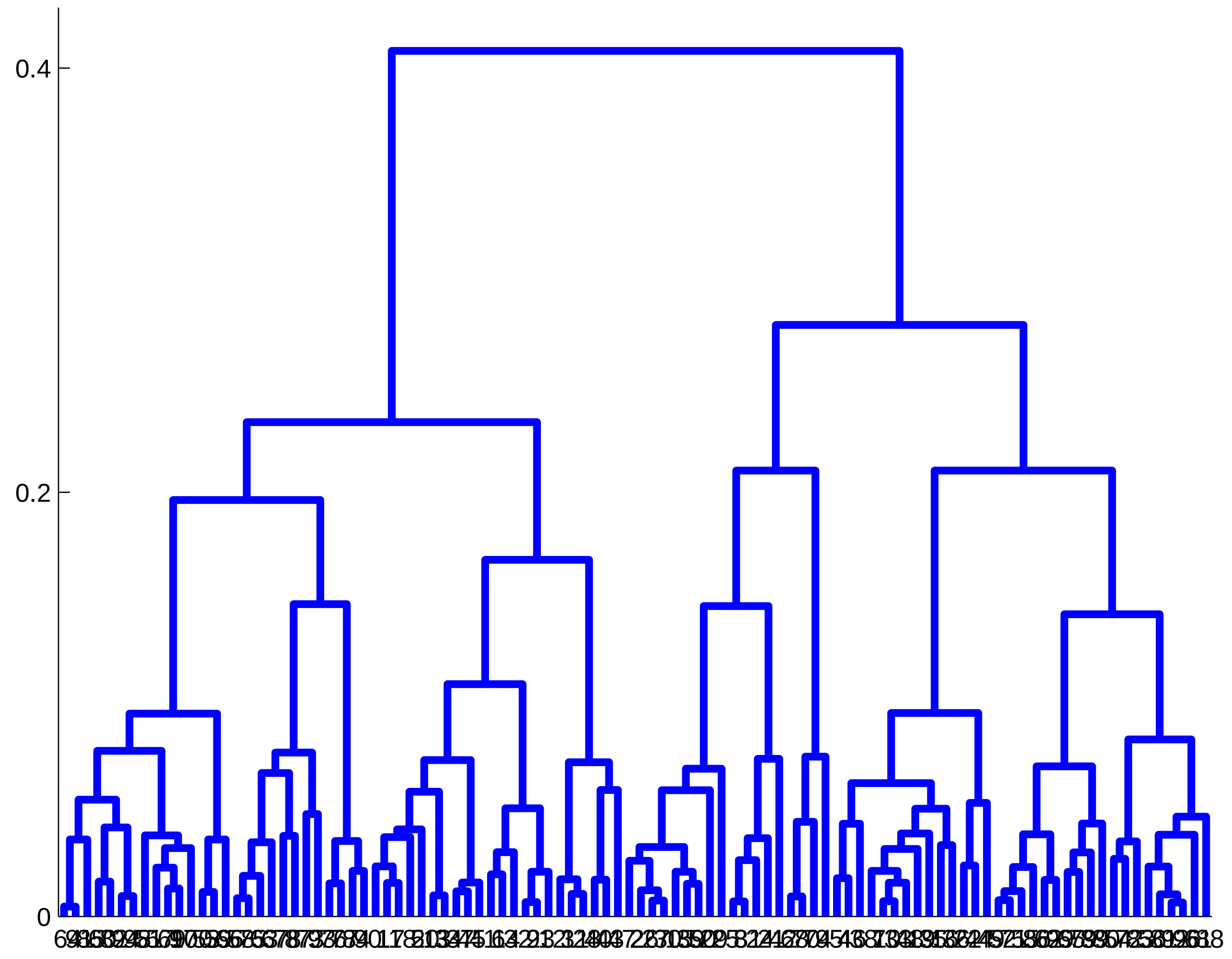
Dendrogram for single-linkage clustering of Example 2



Dendrogram for complete-linkage clustering of Example 2



Dendrogram for average-linkage clustering of Example 2



Some notes

- We can form a flat clustering by cutting of the tree at any height.
- Jumps in the height of the dendrogram can suggest natural cutoffs.

Divisive clustering

- Works by recursively partitioning the data objects.
- Dividing the objects to optimize one of the agglomerative criteria is computationally hard.
- Many heuristics for partitioning data objects have been proposed . . . but many violate monotonicity, making it hard to draw dendrograms.
- (See Hastie, Tibshirani, Friedman p.480 for an example of a divisive clustering algorithm that does have the monotonicity property.)

Segue to dimensionality reduction

Motivation for dimensionality reduction.

- Clustering, flat or hierarchical, can group the data according to similarity, aiding visualization and discovery.
- But we still can't *plot* high-dimensional (or non-numeric) data. (The 2D graphs I've been showing are a bit misleading.)
- Dimensionality reduction (or embedding) techniques:
  - Assign data objects coordinates new coordinates, typically in 2D or 3D.
  - Approximately preserve similarity/distance relationships between objects.
  - Allow us to "see" distance relationships more directly.
- We briefly look at self-organizing maps (SOMs) and multi-dimensional scaling (MDS).

Self-organizing maps

- Assume the data objects are real vectors of length n.
- Try to stretch a “grid” of points in n-dimensional space to approximate the data.
- The grid points are iteratively moved, ”pulled”, by data points, similar to how the centroids of K-means clustering move around.
- The data can then be visualized by mapping each object to the nearest grid point.

Self-organizing maps

- Inputs:
  - A set $D = \{x_1, x_2, \ldots, x_m\}$ of n-dimensional real vectors.
  - A dimension for the grid (1,2 or 3 if we want to plot it.)
  - Number of grid points along each dimension.
- Output: Coordinates G in $\Re^n$ for each grid-point.
  (E.g., for the 2D grid case, $G(i,j) \in \Re^n$ specifies the coordinates of grid-point (i,j).)

One of the simplest SOM algorithms (1D Case)

- Initialize the $G(i)$ somehow.
- Repeat
  - Choose a data point x_l at random.
  - Find the nearest grid-point:

$$i = \arg\min_i \|G(i) - x_l\|$$

  - Find the “neighborhood” of i

$$N(i) = \{(i') : \|i' - i)\| < r$$

  - Move all points in the neighborhood towards x_l:

$$G(i') \leftarrow (1-\alpha)G(i') + \alpha x_l \text{ for all } i' \in N(i)$$

- Typically, $\alpha \longrightarrow 0$ and $r \longrightarrow 1$ over time.

Notes

- If the data approximately lies on a curve or surface, the SOM may capture that structure, but:
  - Different runs can find different solutions.
  - If we try to fit data on a 2D surface with a 1D grid, well. . .
- More sophisticated versions of SOMs use different updating rules.

Multi-dimensional scaling

- SOMs try to stretch a (1D,2D or 3D) grid of points to fit high-dimensional data.
- MDS directly assigns each data object to a point in a low-dimensional Euclidean space so that pairwise distances in that space reflect a given measure of dissimilarity.

Multi-dimensional scaling

- Input:
  - For m data objects, a dissimilarity matrix $\mathcal{DS}$, where $\mathcal{DS}(i, j)$ is the distance between objects i and j.
  - Desired dimension d of the embedding.
- Output:
  - Coordinates $Z(i) \in \Re^d$ for each data object i which (as much as possible) minimizes a *stress function*.
    The stress function quantifies the disagreement between distances specified by $\mathcal{DS}$ and the distances in $\Re^d$.

- Common stress functions include:
  - The least-squares or Kruskal-Shephard criterion:

$$\sum_{i \neq i'} (\mathcal{DS}(i,i') - \|Z(i) - Z(i')\|)^2$$

  - The Sammon mapping:

$$\sum_{i \neq i'} \frac{(\mathcal{DS}(i,i') - \|Z(i) - Z(i')\|)^2}{\mathcal{DS}(i,i')}$$

- Usually, one resorts to a gradient-based optimization to find $Z(i)$ which minimize the stress function.

Summary

- Hierarchical clustering organizes data objects into a tree based on similarity.
  - Agglomerative (bottom-up) tree construction is most popular.
  - There are several choices of linkage criterion.
  - Monotonicity allows us to draw dendrograms in which the height of a node corresponds to the dissimilarity of the clusters merged.
  - Trees can be cut off at some level, to generate a flat partitioning of the data.
- Dimensionality reduction techniques are another way of helping us visualize similarity/distance between data objects.
  - Self-organizing maps stretch a grid to fit high-dimensional data, then project the data onto the grid for low-dimensional viewing.
  - Multi-dimensional scaling directly maps data objects into a low-dimensional space, trying to preserve dissimilarities.