CMPINF 2100

Introduction to Data Centric Computing

Week 09

Cluster analysis: Continued

We used KMEANS to cluster rows based on the columns. The major aspects of KMEANS that you MUST remember are:

- You **MUST** specify the number of clusters **BEFORE** running the algorithm!
  - `n_clusters` argument to `KMeans()`
- Algorithm **ITERATES** from a starting guess. You must make sure **ENOUGH** iterations are used.
  - `max_iter` argument to `KMeans()` specifies the maximum number of iterations to use.
- We do NOT know the best possible initial guess. We RANDOMLY create the initial guess! **RESTART** KMeans from **MANY** different RANDOM initial guesses!
  - `random_state` argument to `KMeans()` sets the random seed to control randomness.
  - `n_init` argument to `KMeans()` specifies the number of random restarts! I recommend at least 25 random restarts.

We used KMEANS to cluster rows based on the columns. PRACTICAL considerations when applying KMeans

- KMeans **CANNOT** work with **MISSING** entries.
  - You MUST remove missings BEFORE applying KMeans.

- Similar rows are clustered together. Similarity is based on **distance** which is based on the VALUES within the COLUMNS.
  - A column with VERY large magnitude and scale compared to the other columns will DOMINATE the distance calculation.
    - The clusters will be based on that variable!
  - A column with very TINY magnitude and scale compared to other columns will essentially be IGNORED by the distance calculation.
    - The clusters will ignore that variable even if you think it is important!

- Therefore, you MUST **PRE-PROCESS** the columns to remove MAGNITUDE and SCALE effects.
  - A common approach is to **STANDARDIZE** the columns
  - You can use `StandardScaler()` from the `sklearn.preprocessing` module.

Limitations of KMeans

- Clustering based on averages, so results can be sensitive to extreme values.

- Cannot handle discrete or categorical variables.

- KMeans does **NOT** tell you the **OPTIMAL** number of clusters to use!
  - Can decide the BEST number of clusters by calculating the TOTAL WITHIN SUM OF SQUARES for MANY possible number of clusters.
    - The `.inertia_` attribute associated with a fitted `KMeans()` object stores the TOTAL WITHIN SUM OF SQUARES.
  - Need heuristics like the KNEE-BEND plot to then decide which clustering result is "best". No right answer…but some results are more appropriate than others.

Let's now see an approach that does **NOT** require us to specify the number of clusters!

- Hierarchical clustering is a technique that contains clusters **WITHIN** clusters!

- The hierarchical structure effectively allows considering many possible numbers of clusters!

- We decide the appropriate number of clusters **after** running the algorithm.

Hierarchical clustering contains clusters within clusters

- The clusters can be created two ways: **Agglomerative** and Divisive

- **Agglomerative** starts with as many clusters as observations.
  - The most similar clusters are fused iteratively, building up the **TREE** in a **bottom-up** fashion.
  - The algorithm ends when all observations are contained in a single large cluster.

Hierarchical clustering contains clusters within clusters

- The clusters can be created two ways: Agglomerative and **Divisive**

- **Divisive** is the opposite. Starts with all observations contained in one large cluster.
  - The algorithm proceeds to break up that cluster until all observations correspond to their own cluster.
  - **Top-down** algorithm.

Agglomerative vs Divisive

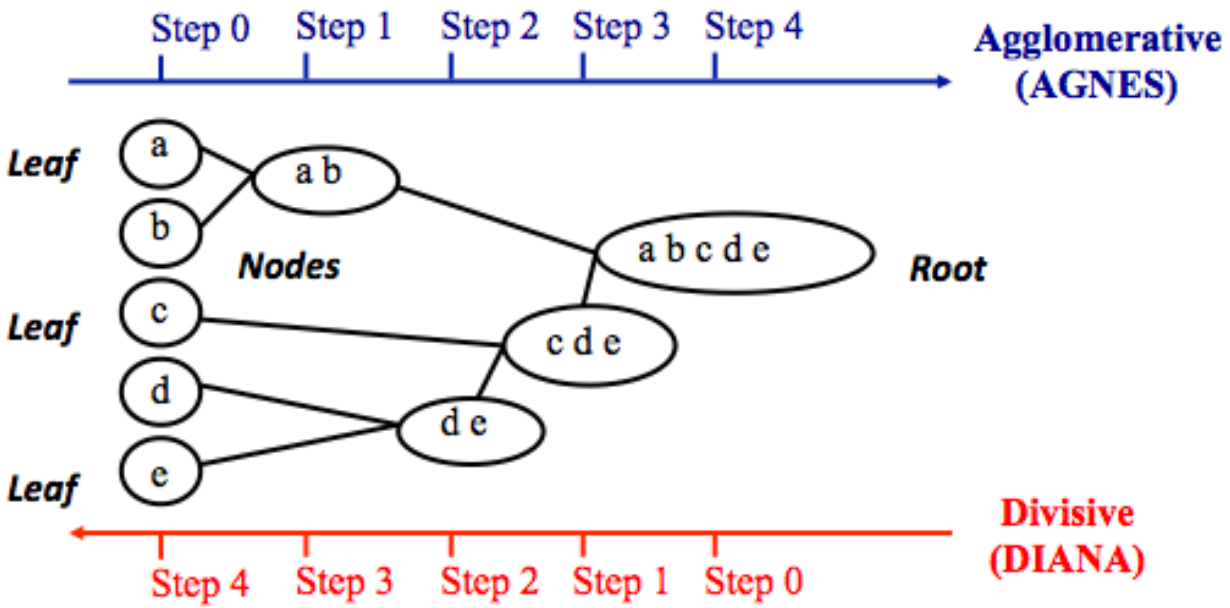
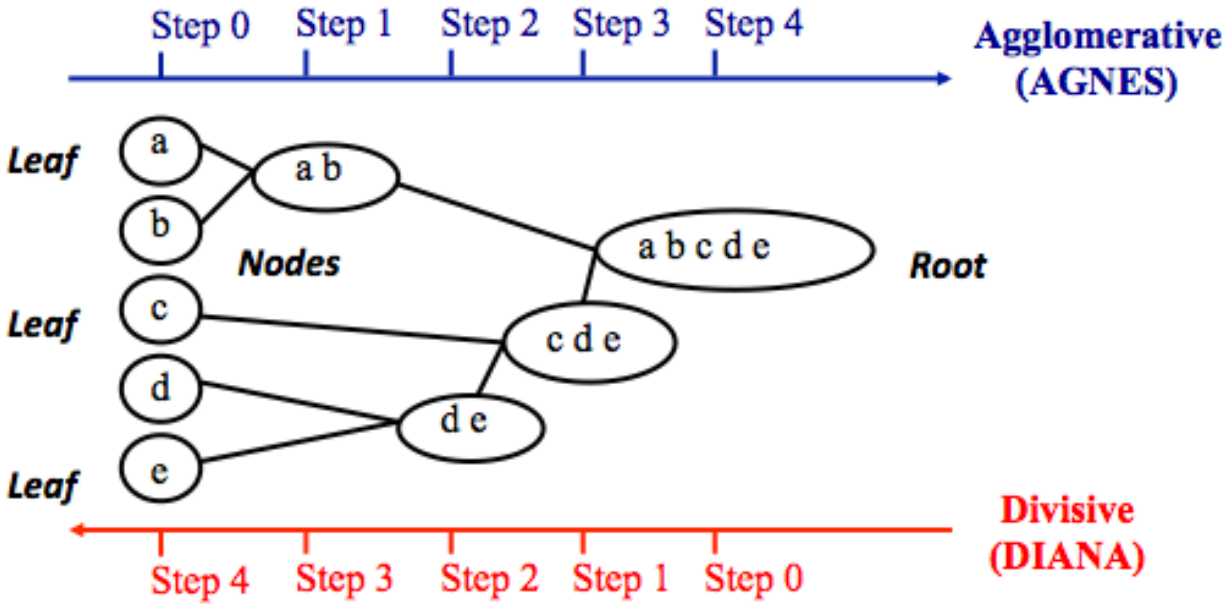


Image from: https://uc-r.github.io/hc_clustering

Agglomerative vs Divisive



We will focus on Agglomerative Nesting (AGNES)

Image from: https://uc-r.github.io/hc_clustering

How does hierarchical clustering define similar?

- Just as with KMeans, similarity is based on **distance**.

- Euclidean distance is a popular choice.

- Therefore, columns MUST be **PRE-PROCESSED** just as with KMeans.
  - We need to **REMOVE** the MAGNITUDE and SCALE effect!
  - Standardization is a popular approach!

How does hierarchical clustering define similar?

- However, compared to KMeans, hierarchical clustering has 2 types of similarity.

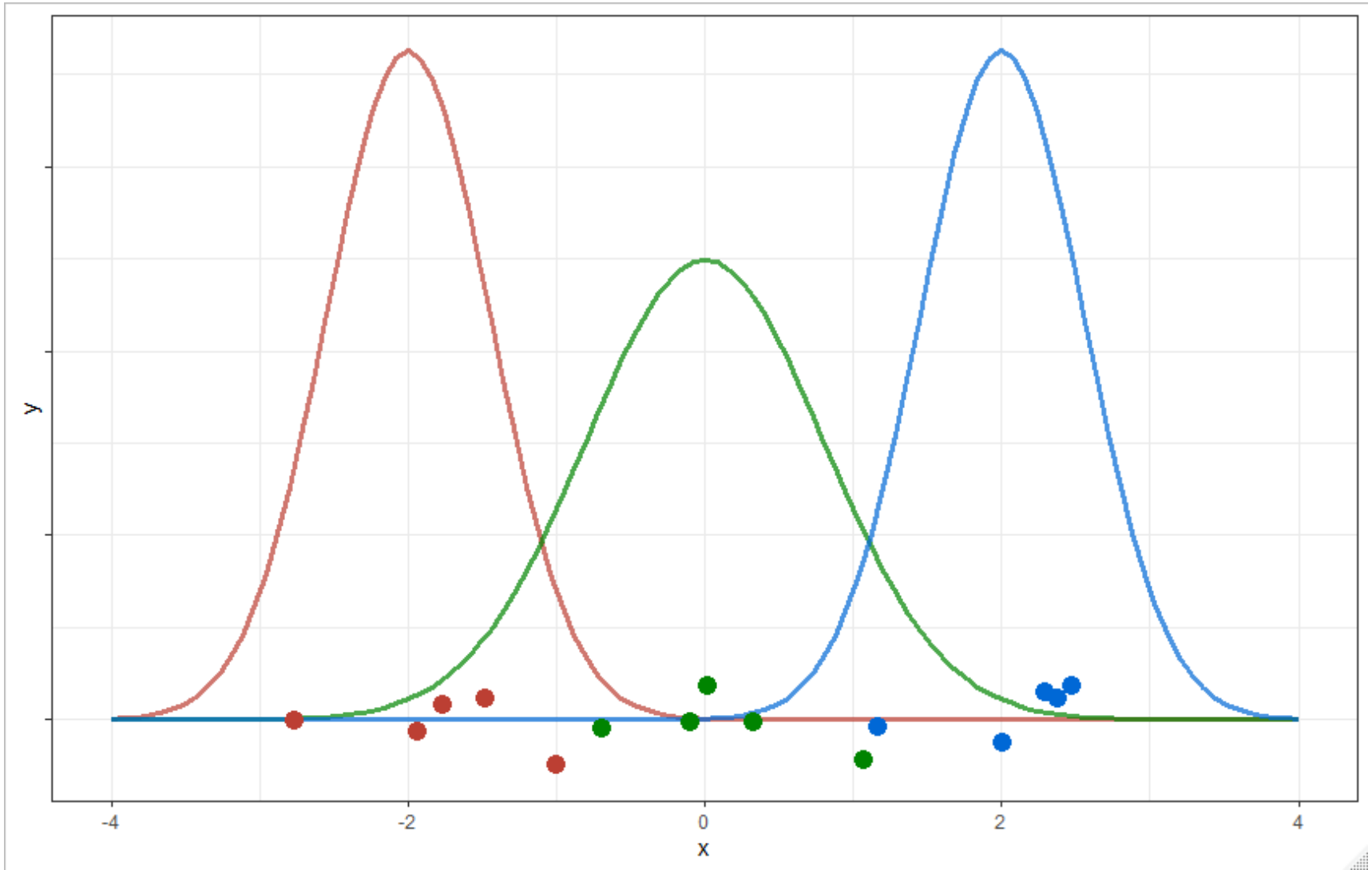
- First type is the similarity between observations.
  - This is the SAME similarity used in KMeans.

- Second type is the similarity between clusters!
  - This does NOT exist in KMeans!!!!

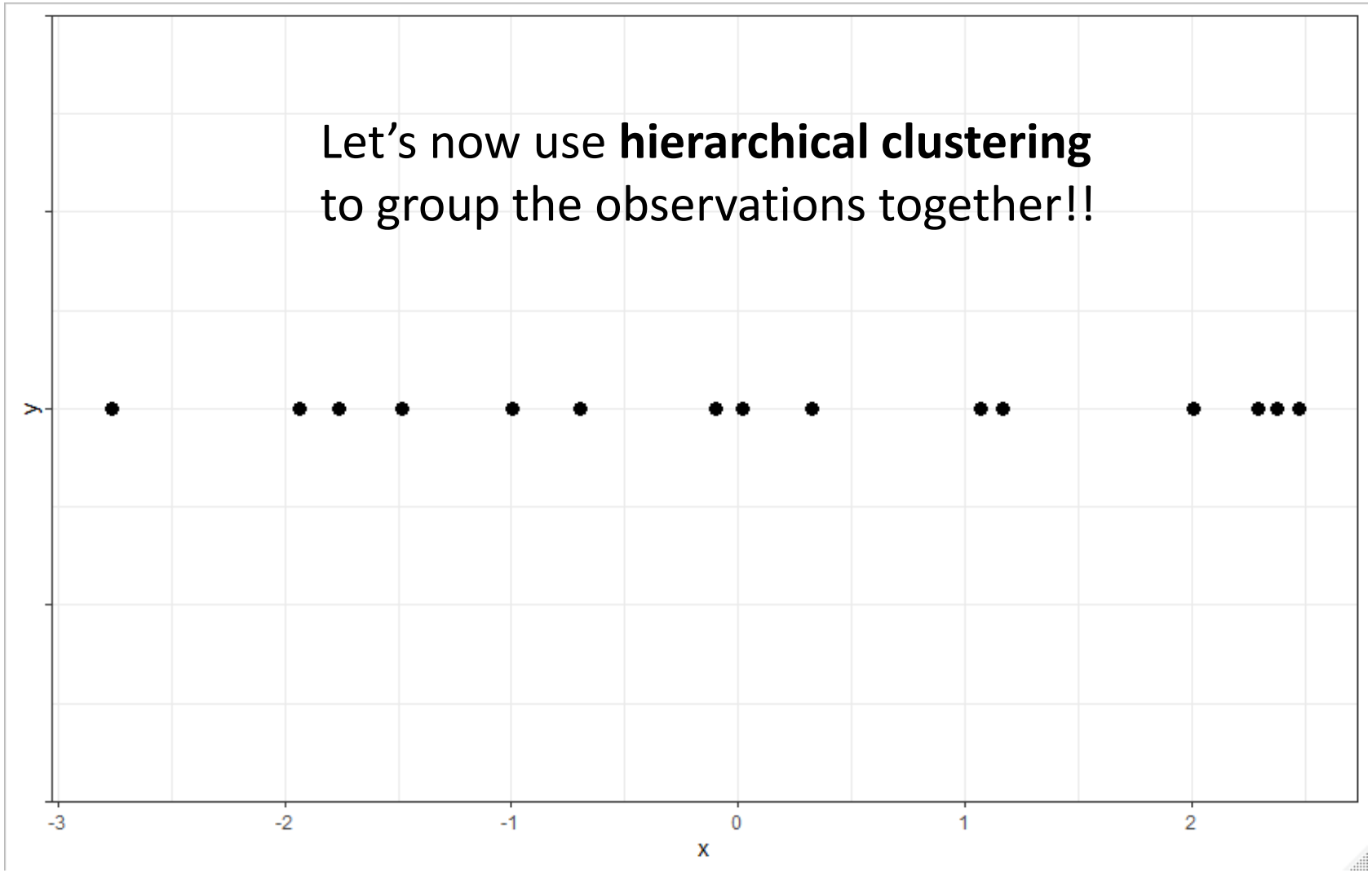
To understand the differences between the two let's return to the simple 1D problem from last week.
As a reminder there are 15 observations randomly generated from 3 Gaussians.

The 3 “real” Gaussians that generated the data are shown to the right.

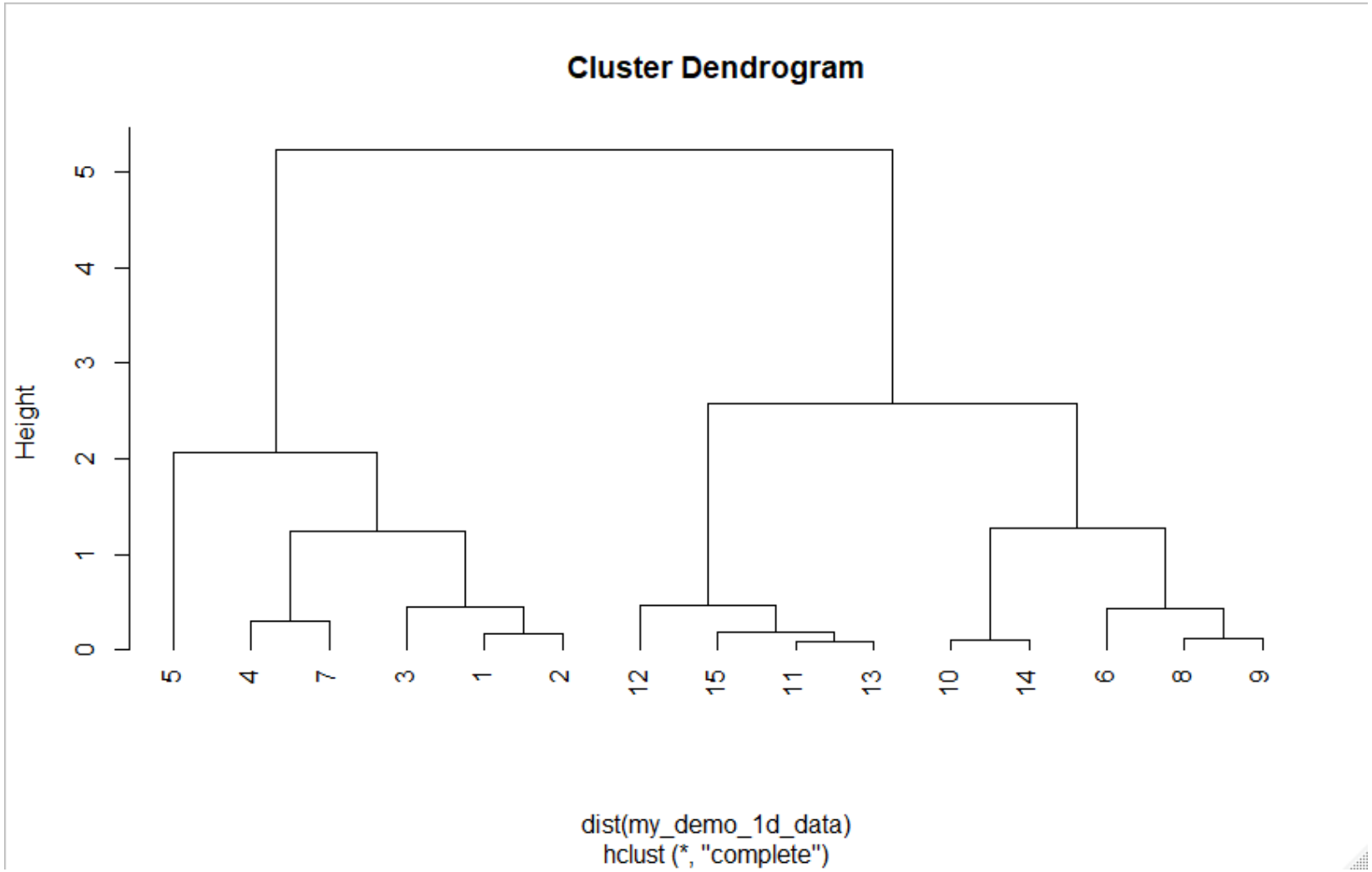
The marker color denotes the observations associated with each Gaussian.



But let’s pretend again we do NOT know the REAL groupings!
We want the clustering method to IDENTIFY the groups!

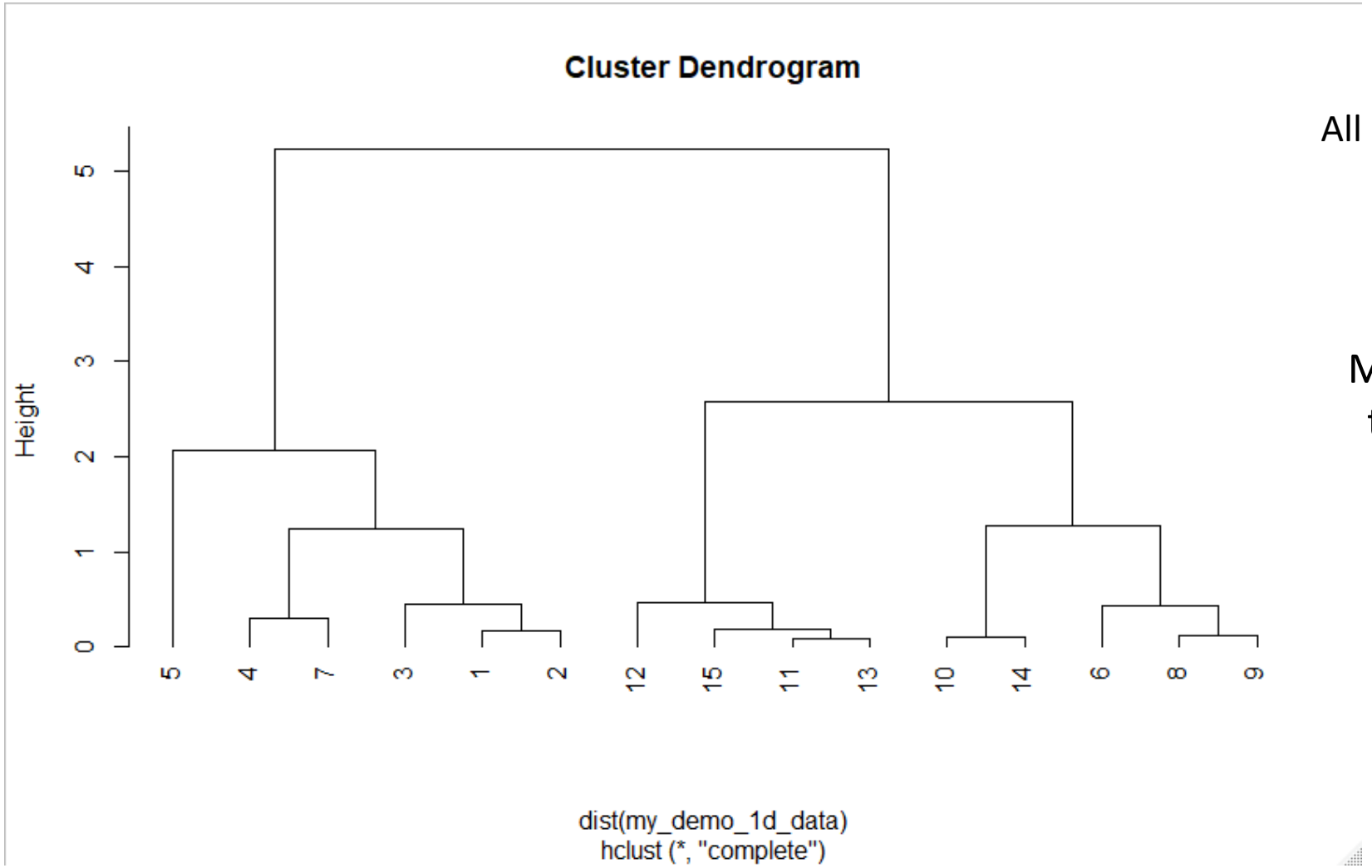


Hierarchical clustering results are presented in a **TREE** like picture called a **Dendrogram**.



How to read a Dendrogram:

Dissimilarity increases with vertical height

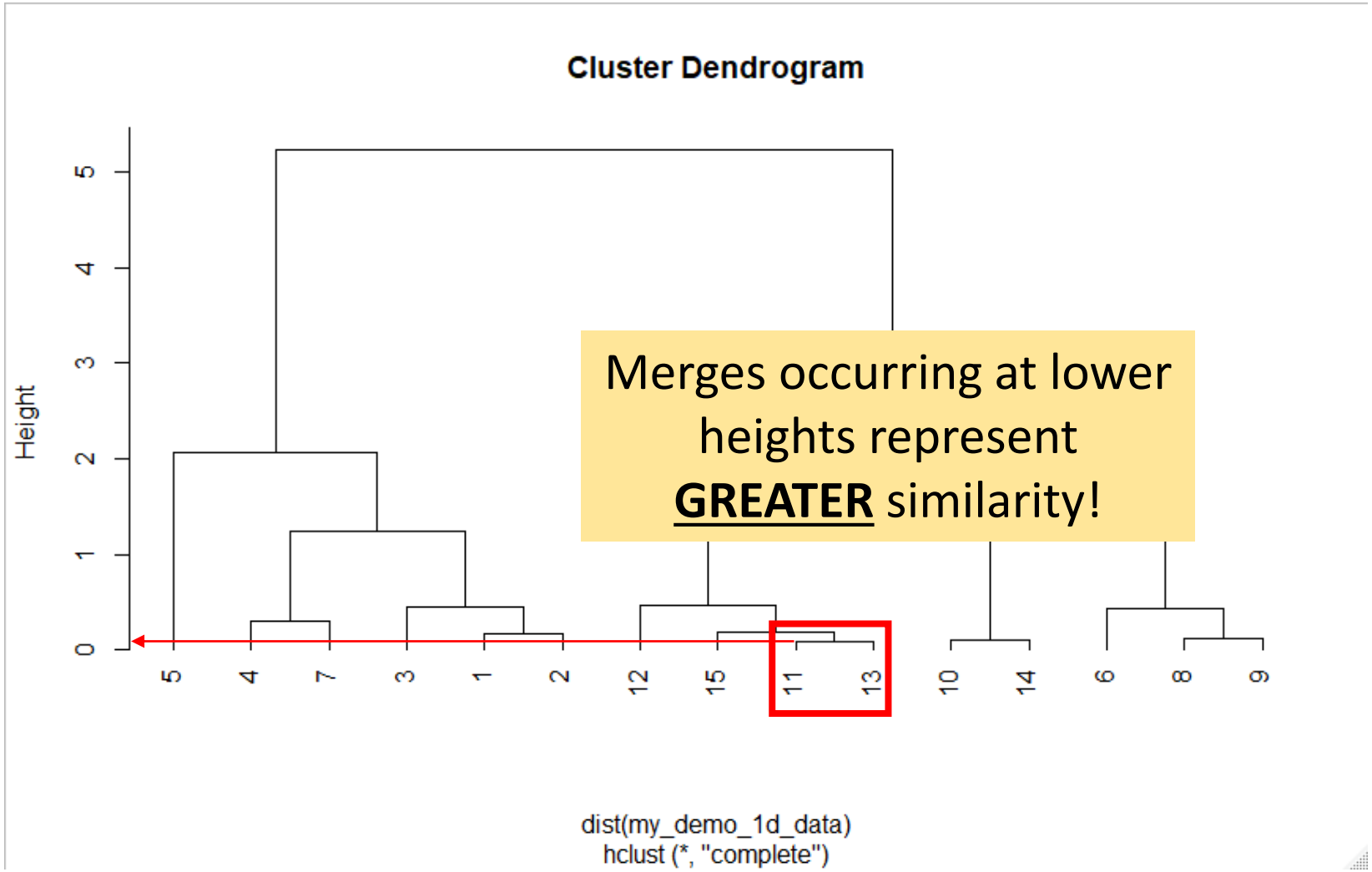


All observations belong to a single cluster

Merge observations together and fuse clusters

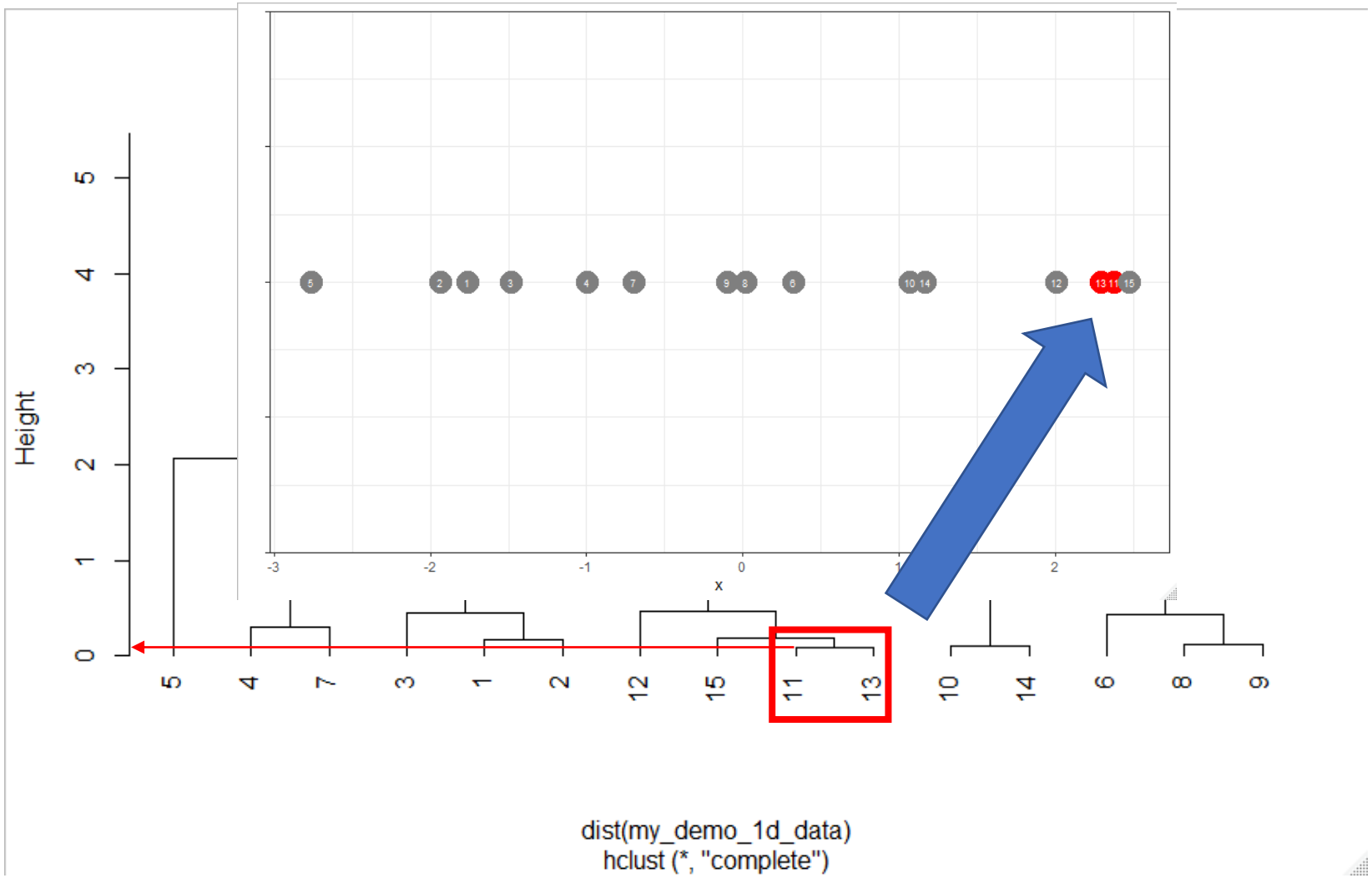
Observations are their own clusters

Vertical axis tells you the similarity

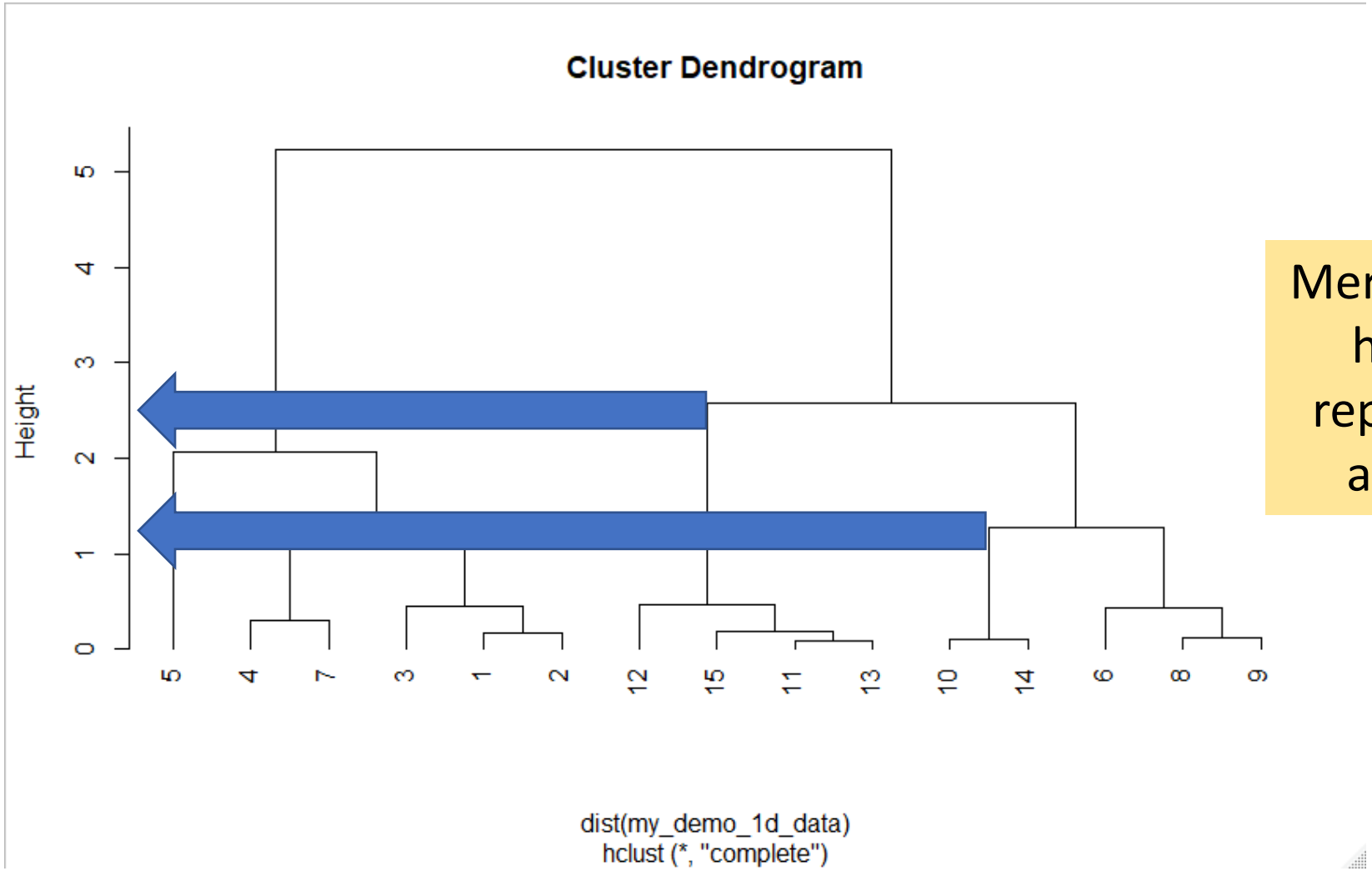


Merges occurring at lower heights represent **GREATER** similarity!

Low dissimarility (high similarity) points are close together!

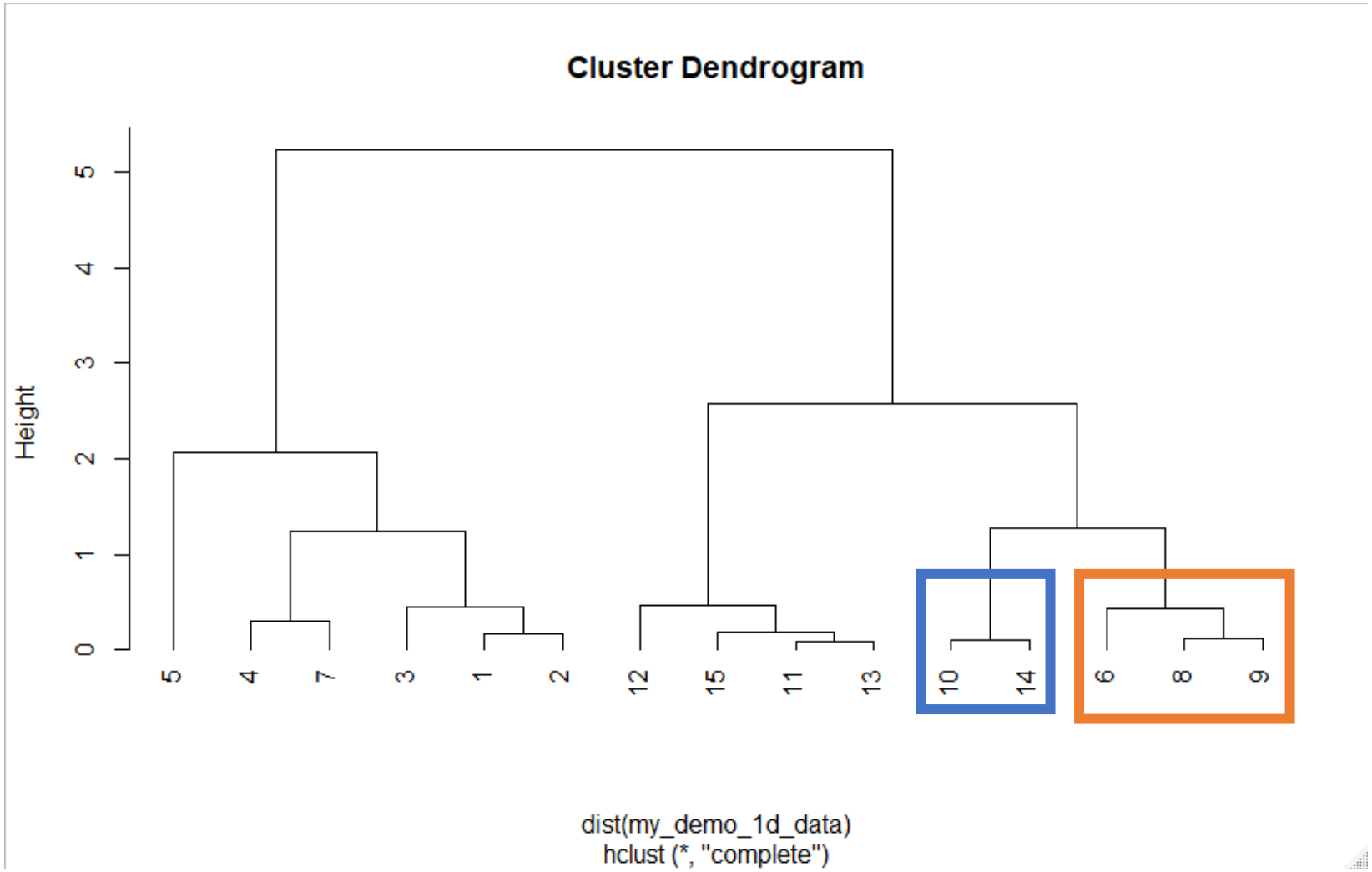


Move up the dendrogram

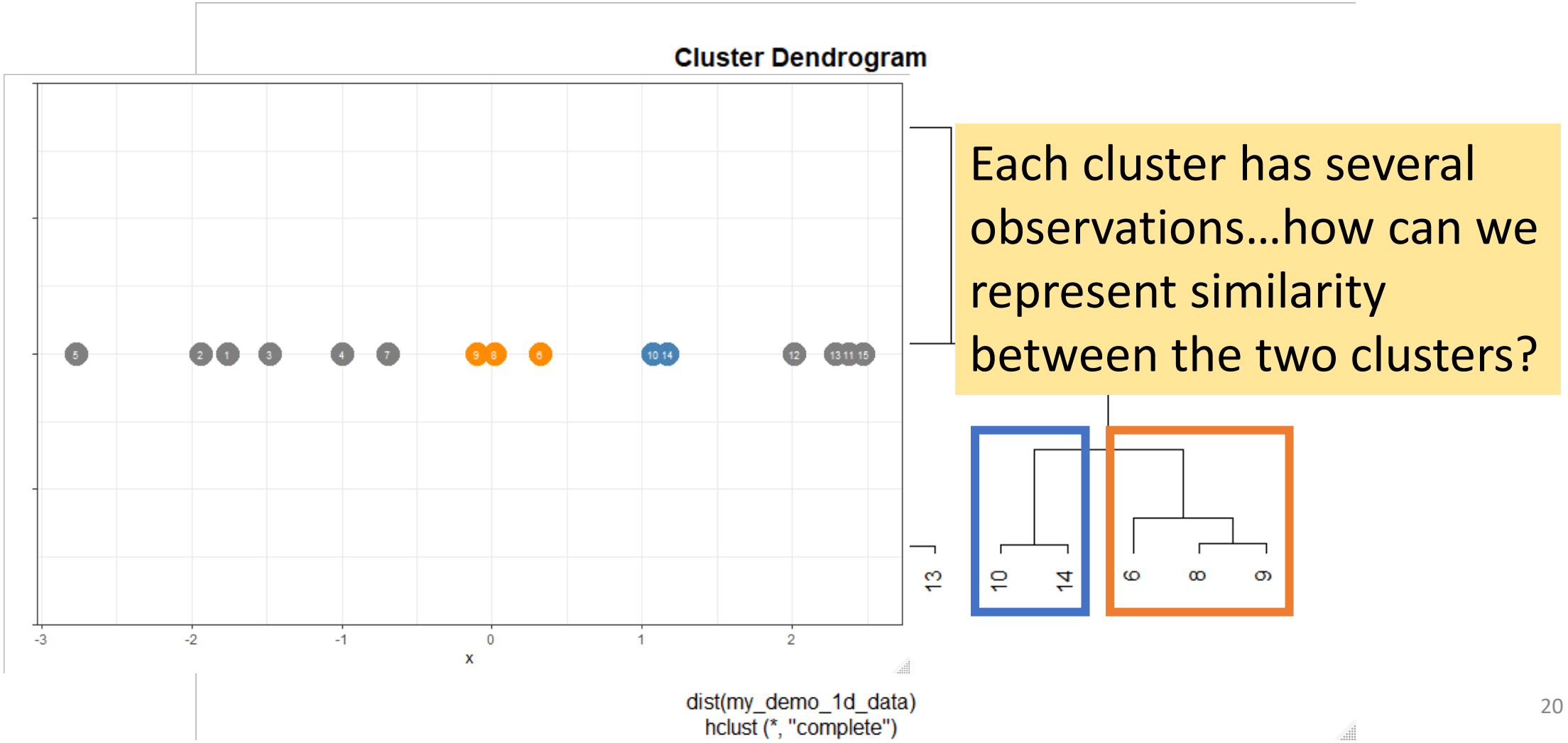


Merges occurring at higher heights represent clusters are less similar

How are clusters fused together?

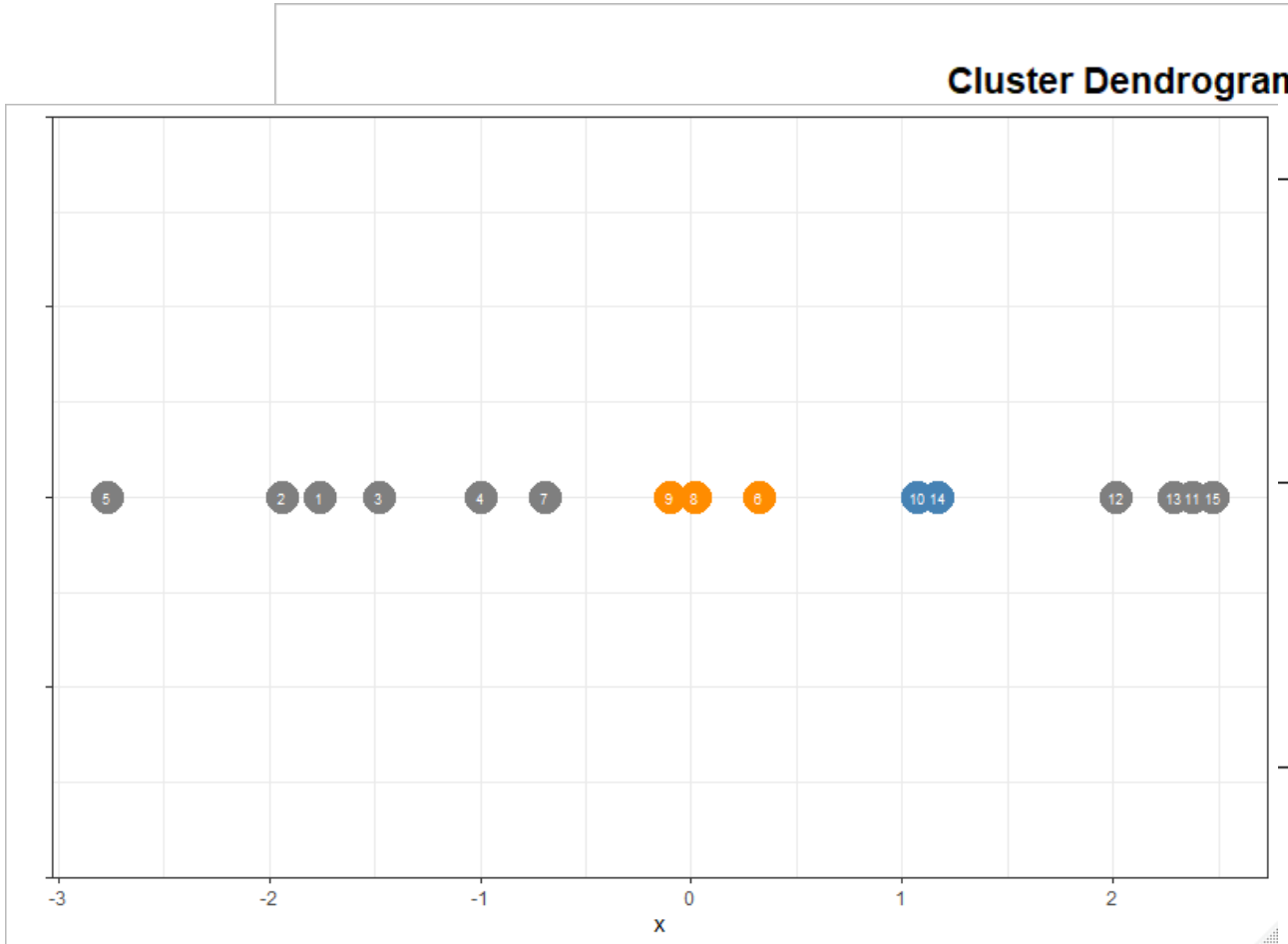


How are clusters fused together?

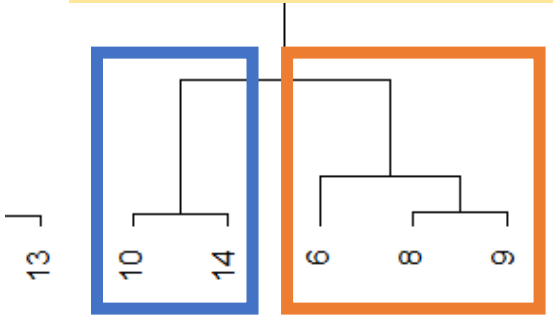


Each cluster has several observations…how can we represent similarity between the two clusters?

How are clusters fused together?



Each cluster has several observations…how can we represent similarity between the two clusters?



There are DIFFERENT ways to measure cluster similarity!!!!

dist(my_demo_1d_data)
hclust (*, "complete")

Measuring cluster similarity is referred to as the LINKAGE function.

We need to specify the LINKAGE between the clusters to measure their similarity.

- **COMPLETE** or maximum linkage:
  - Compute all pairwise dissimilarities between the two clusters.
  - Take the **MAXIMUM** value as the distance between the two clusters.

- **SINGLE** or minimum linkage:
  - Compute all pairwise dissimilarities between the two clusters.
  - Take the **MINIMUM** value as the distance between the two clusters.

We need to specify the LINKAGE between the clusters to measure their similarity.

- **AVERAGE** or mean linkage:
  - Compute all pairwise dissimilarities between the two clusters.
  - Compute the **AVERAGE** of the dissimilarities and use that as the distance.

- **CENTROID** linkage:
  - Calculate the CENTROIDS associated with each cluster.
  - Compute the dissimilarity between the clusters based on the distance between their centroids.

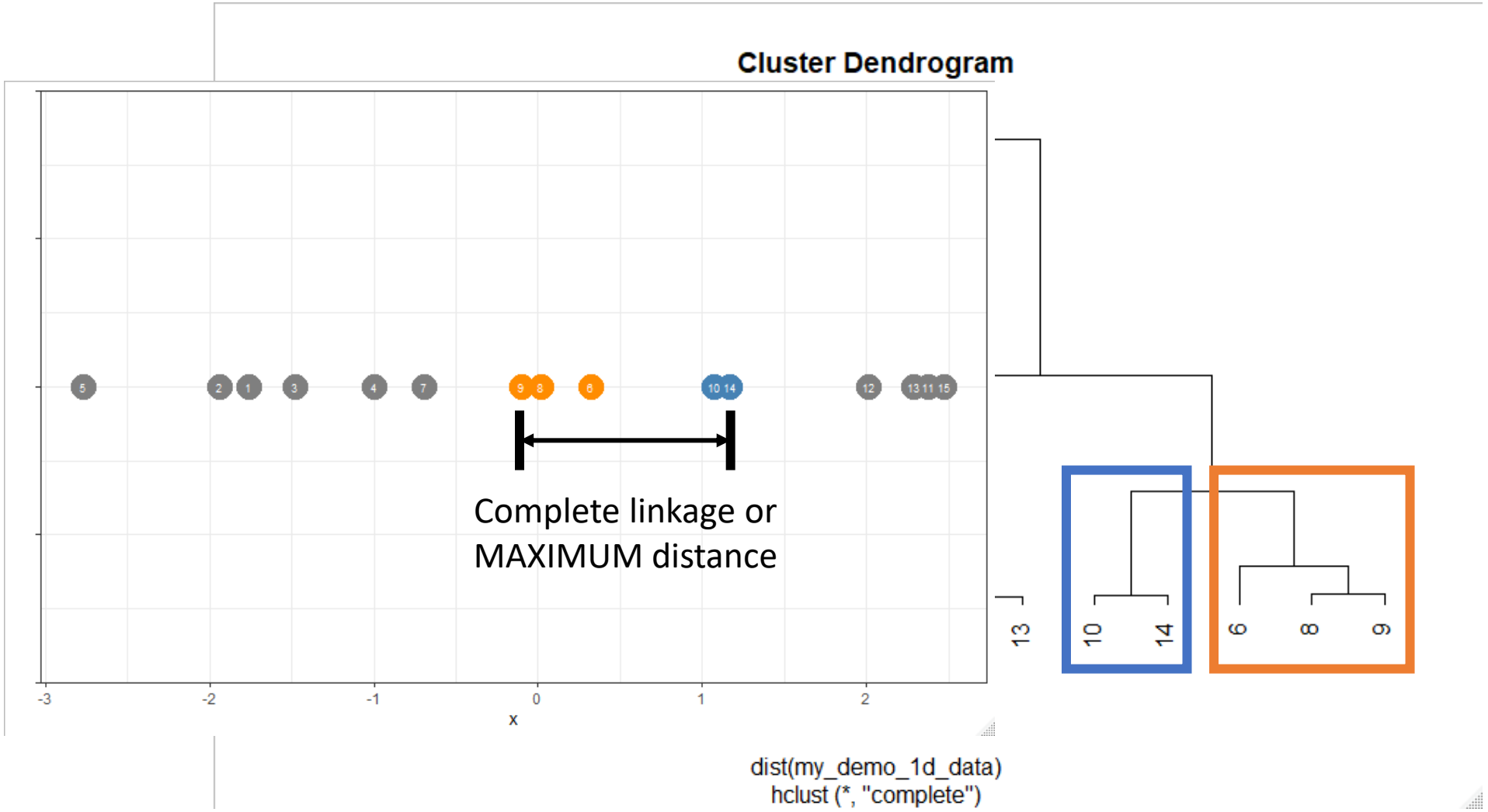
We need to specify the LINKAGE between the clusters to measure their similarity.

- **Ward**:
  - Minimize within cluster variance.
  - At each step in the algorithm, fuse the clusters with the minimum between cluster distance.

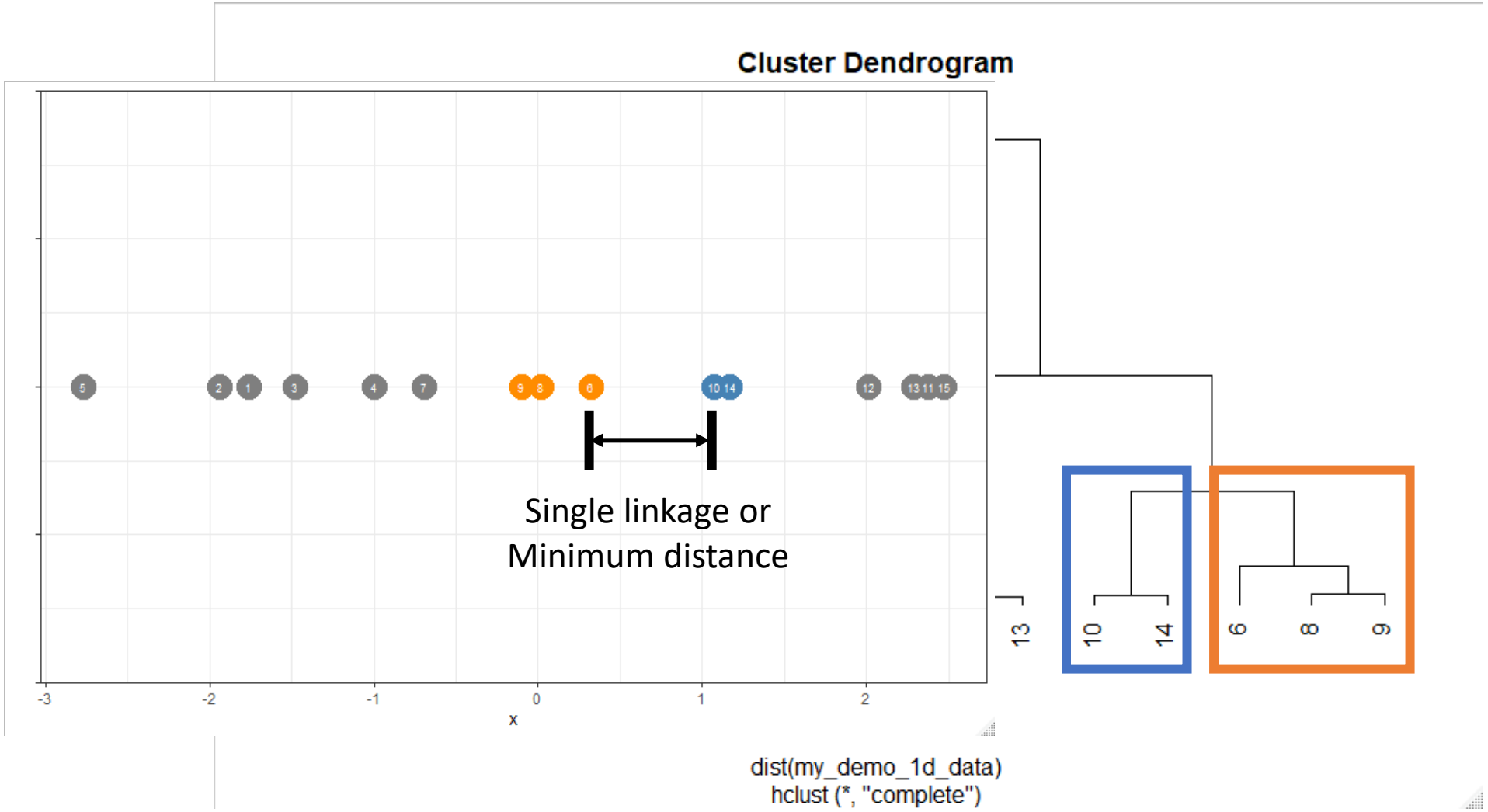
The Ward method is my preferred approach.

I view it as trying to optimize the clusters at each step of the algorithm.

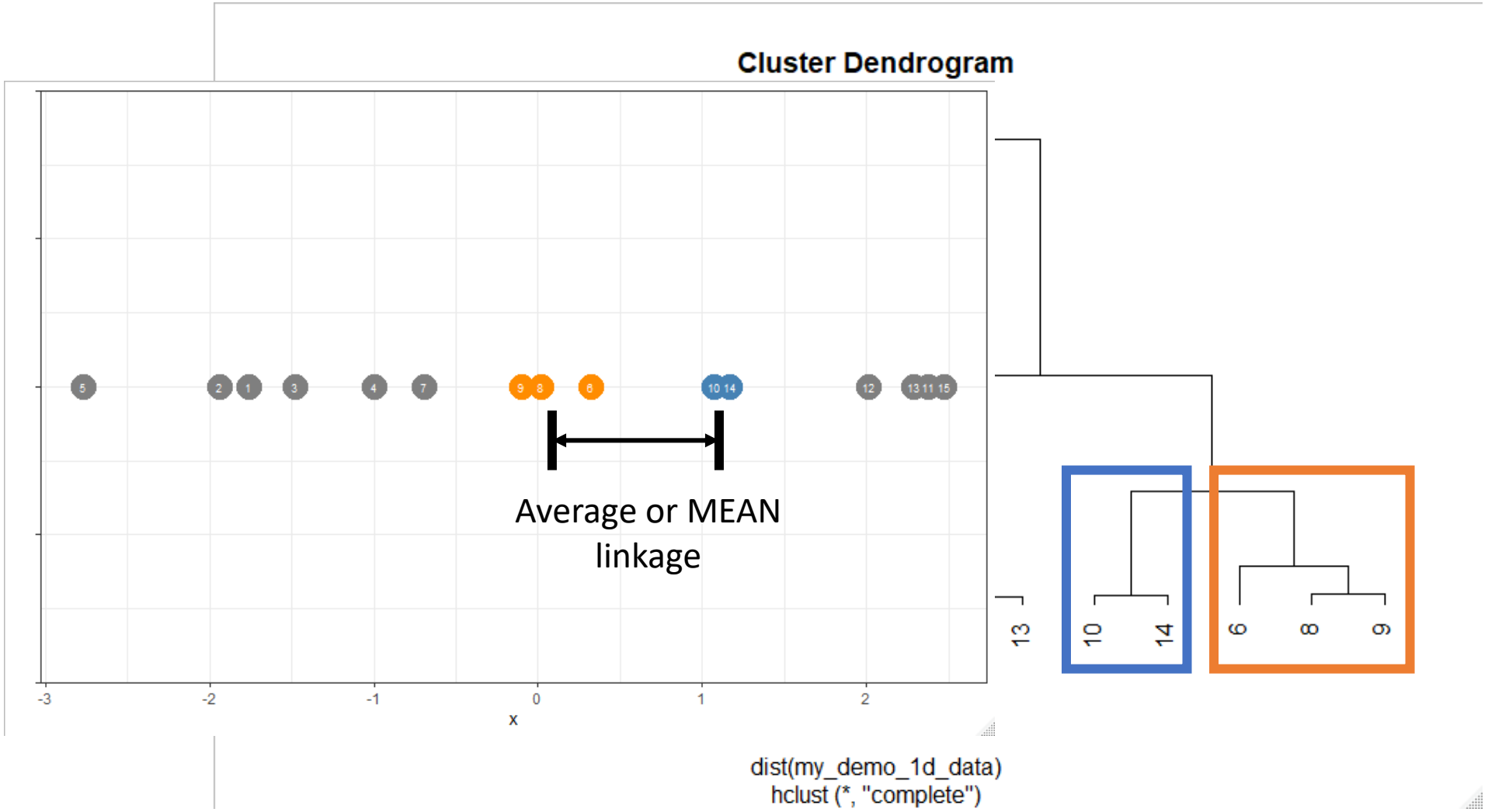
COMPLETE linkage is easy to visualize.
It is the MAXIMUM distance between two clusters.



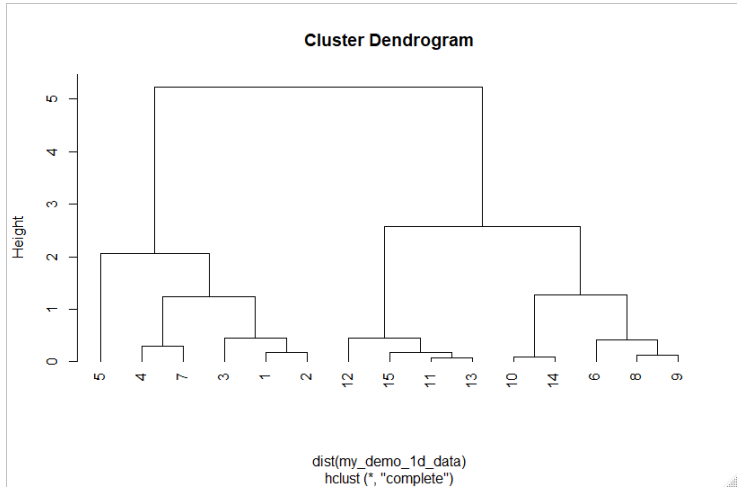
SINGLE linkage is also easy to visualize. It is the SHORTEST distance between two clusters.

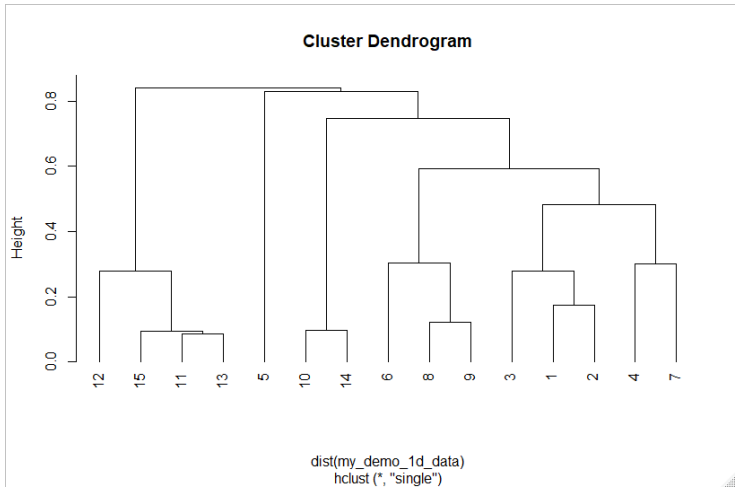


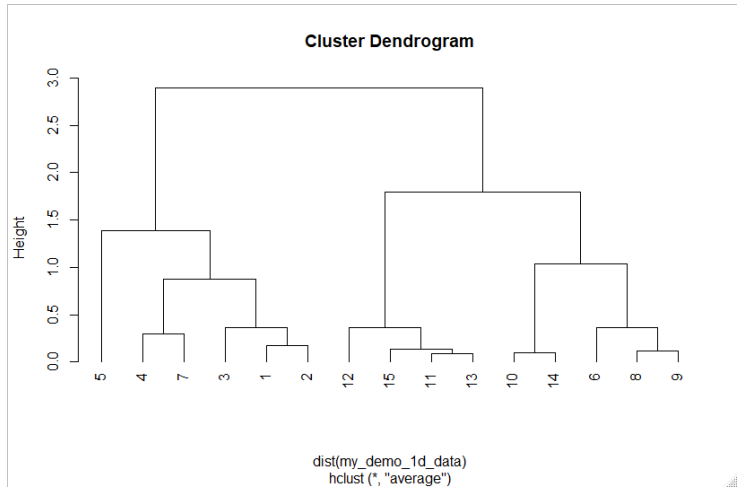
The **AVERAGE** linkage is the AVERAGE distance between the two clusters.

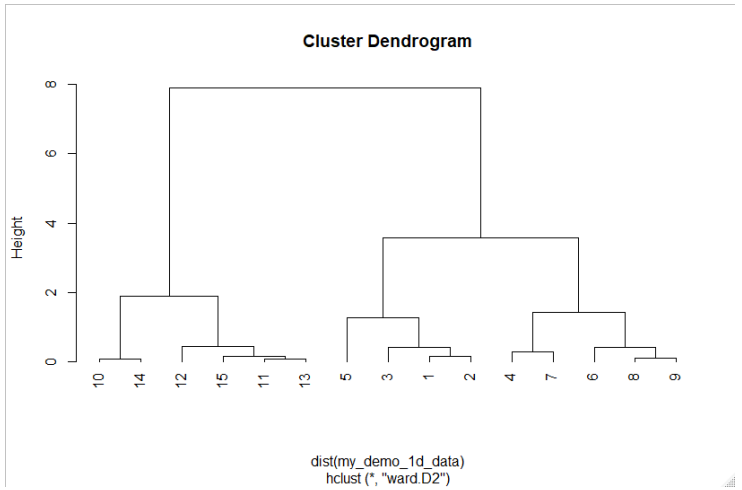


Different linkages produce different distances, the cluster results can be substantially different!!



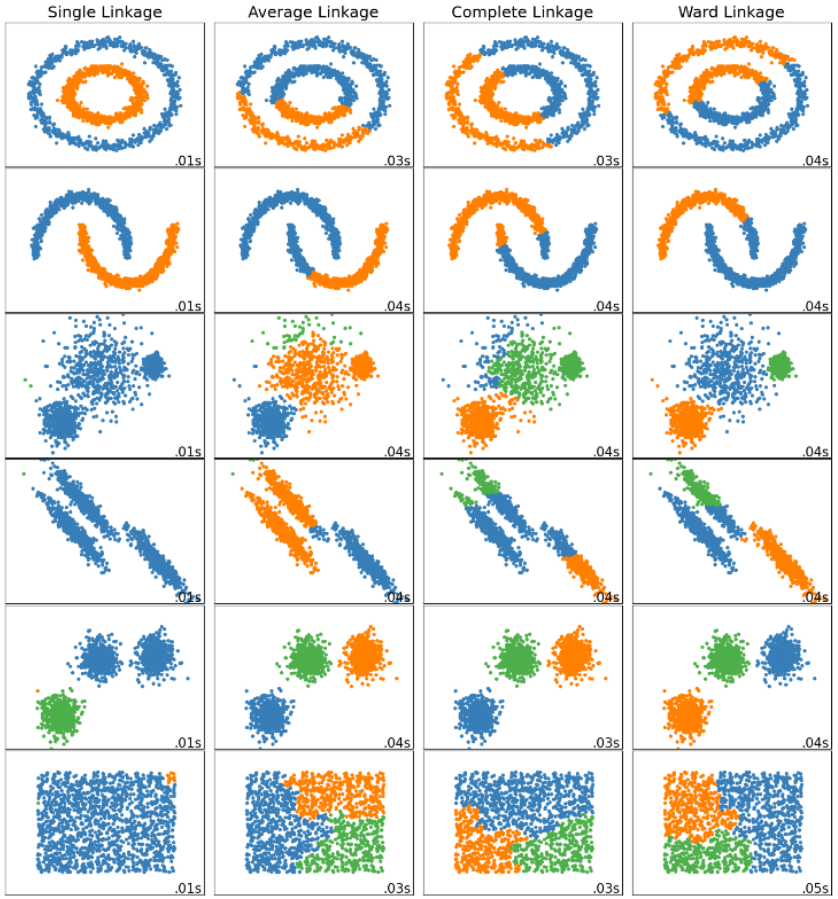






Scikit-Learn has a nice example demonstrating the differences between several linkages

- Scikit-Learn example page link.
- You do not need to worry about the code shown in the example.
- The figure shown at the bottom of the page (reproduced on the right) is what matters.

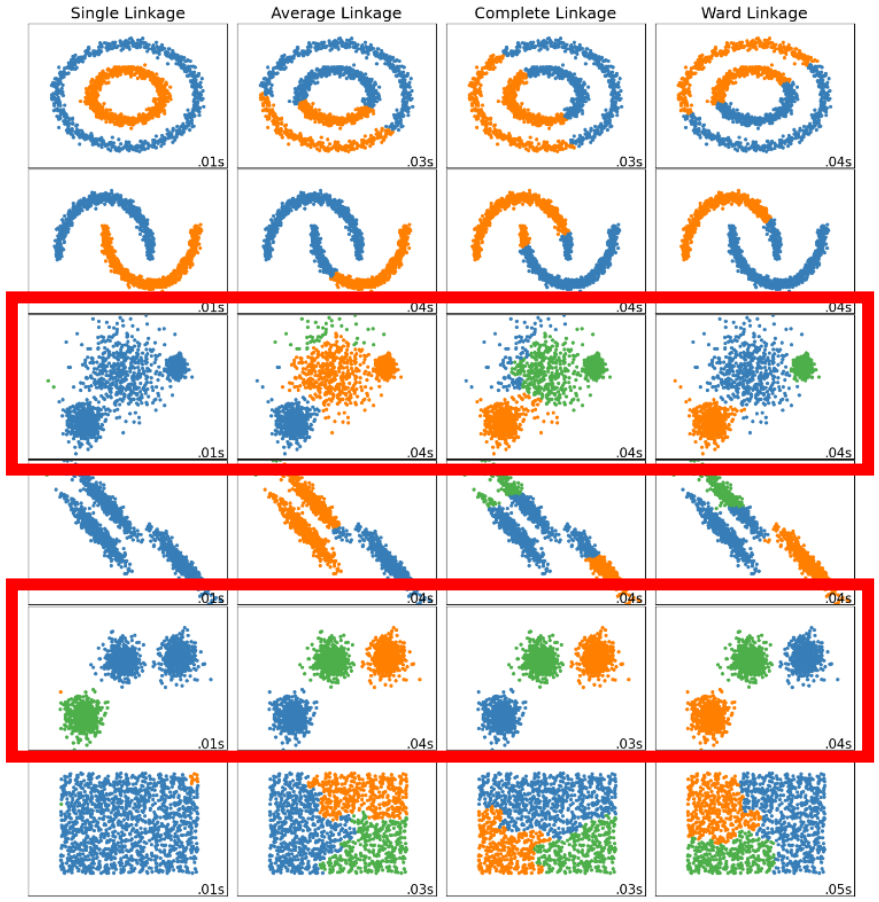


Scikit-Learn has a nice example demonstrating the differences between several linkages

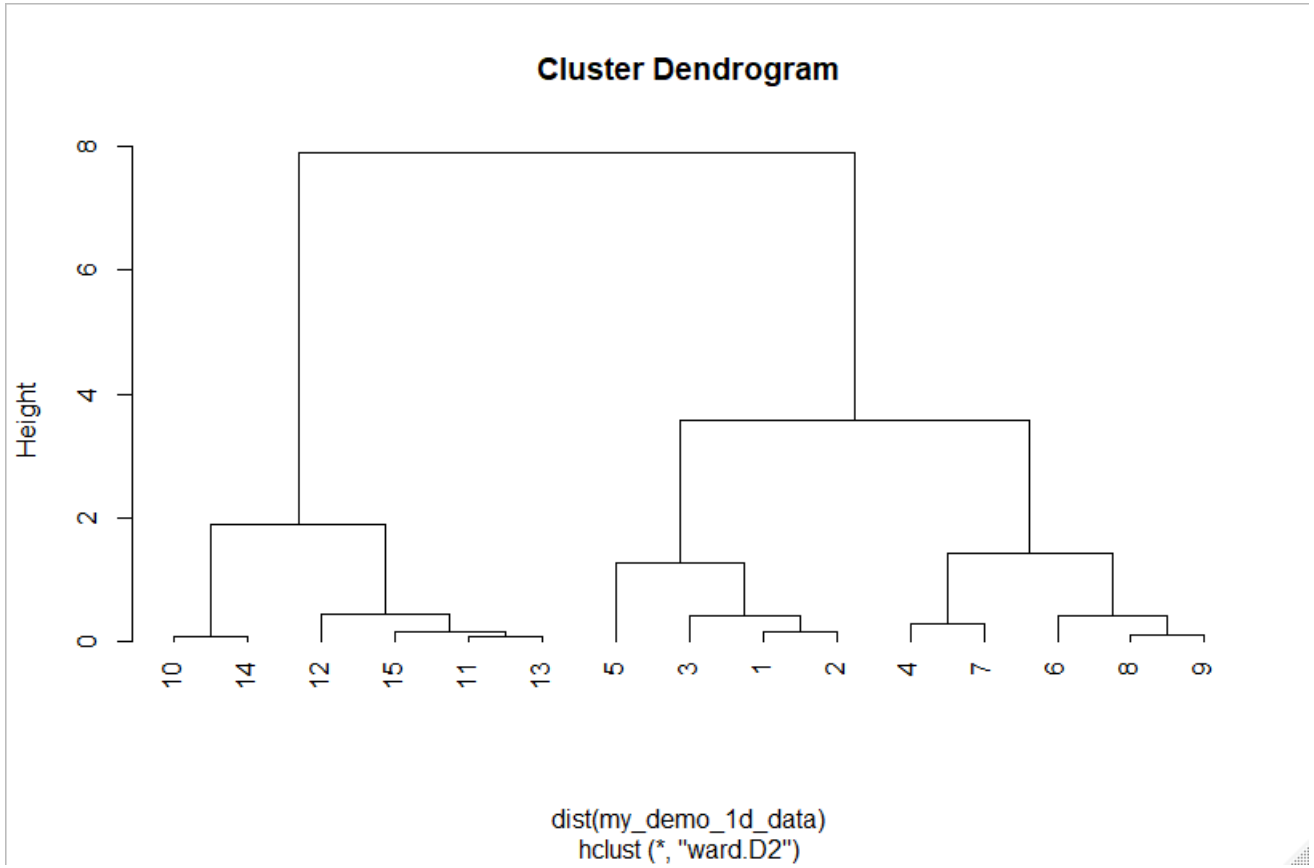
- [Scikit-Learn example page link](#).

- Ward method works well at separating the observations into separate “blobs”.

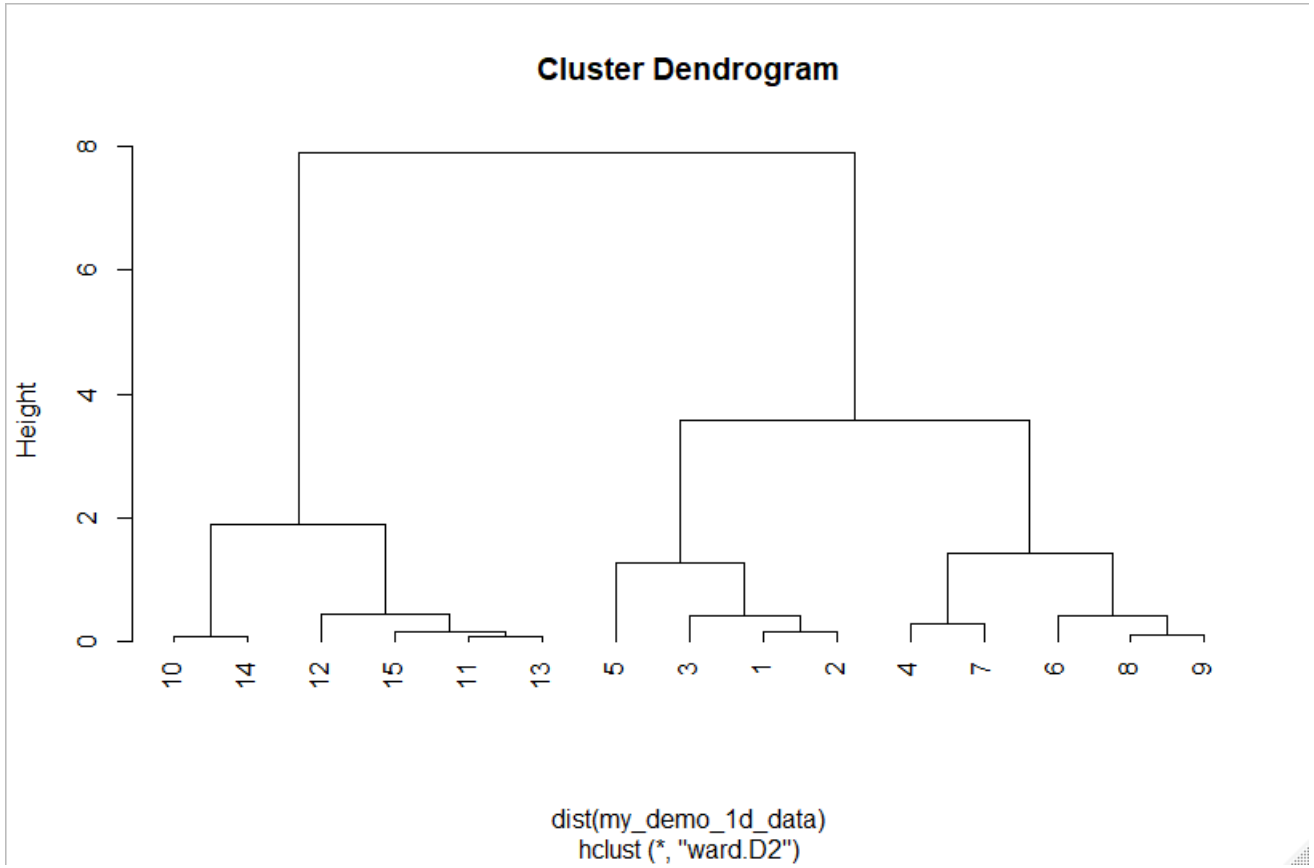
- The “blobs” can be different sizes.



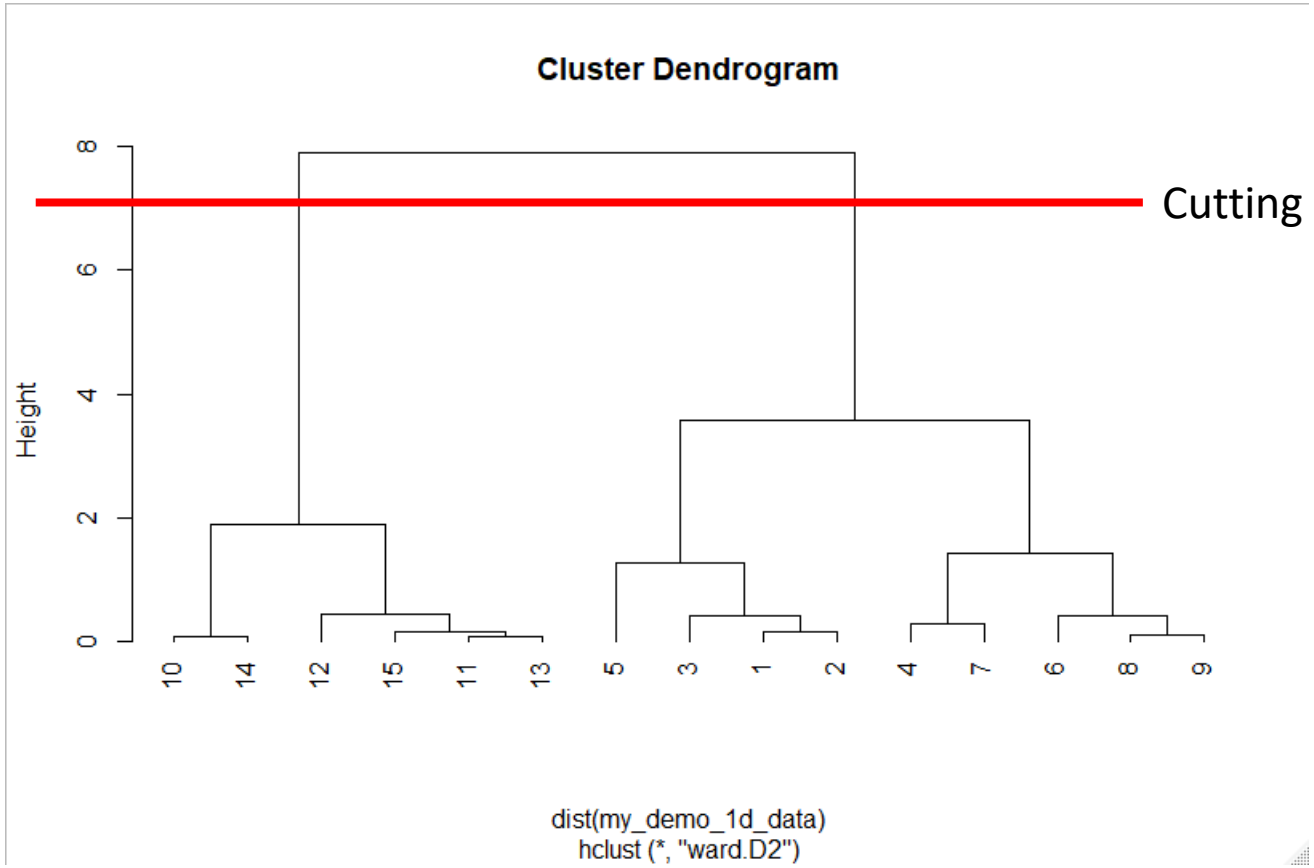
Cluster assignments are made by “cutting the tree”



Which height should we cut the tree?

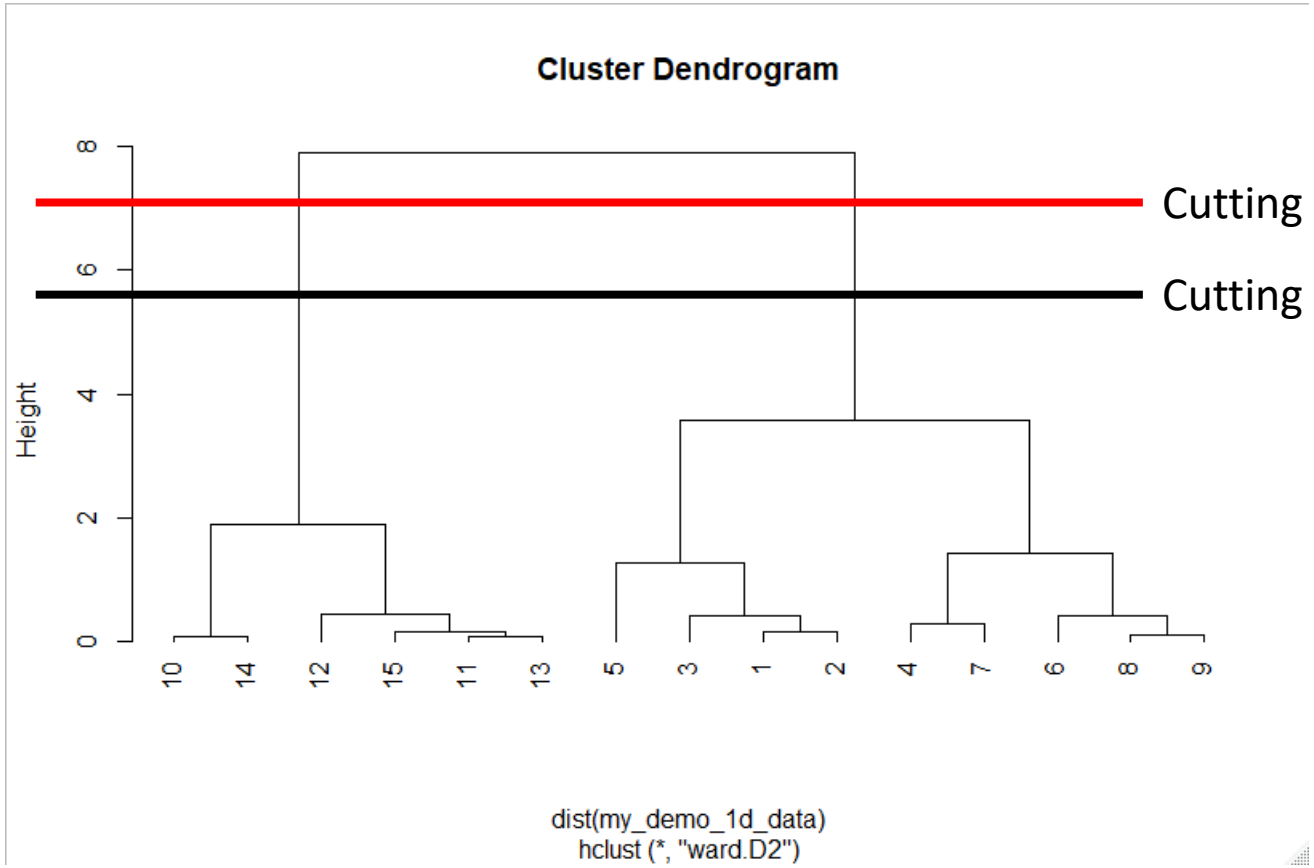


Which height should we cut the tree?

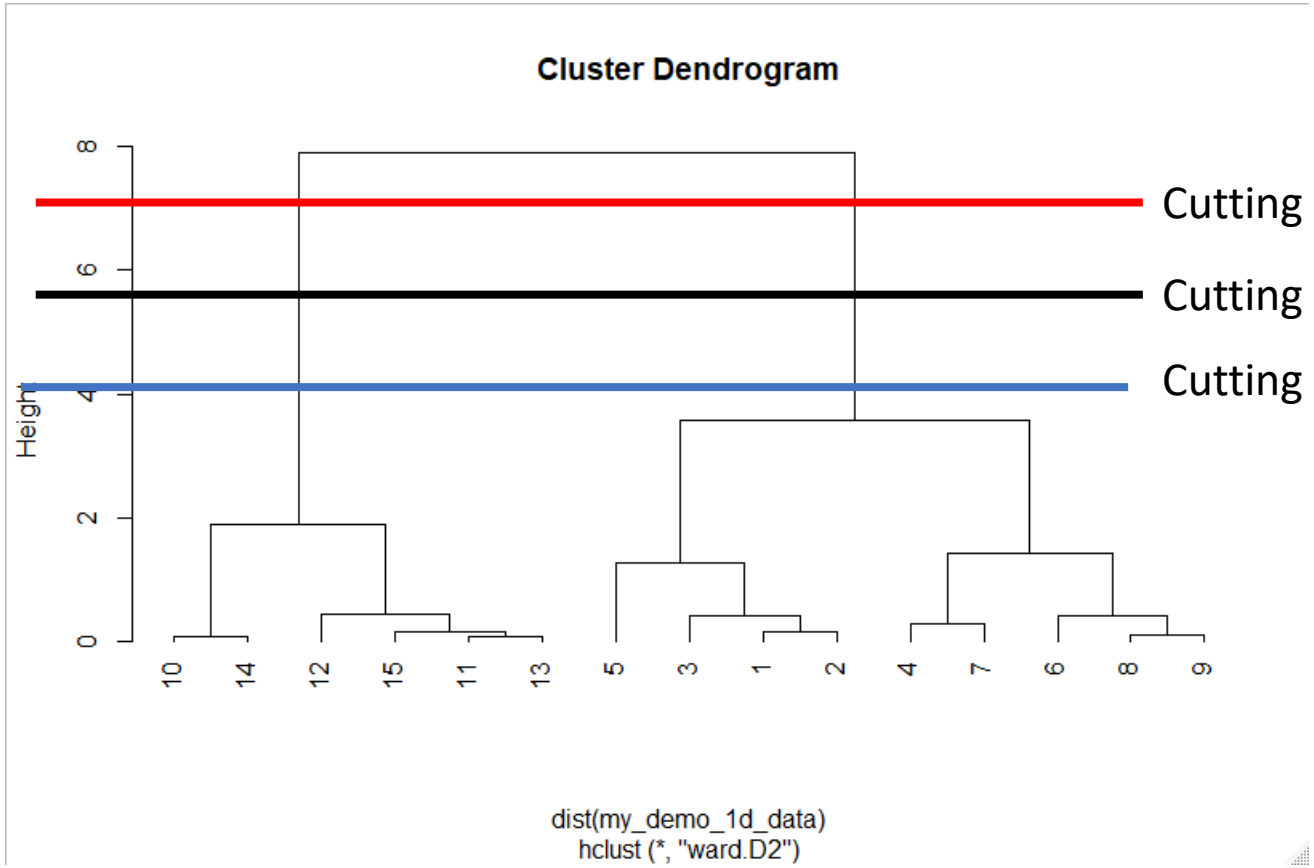


Cutting here produces 2 clusters

Which height should we cut the tree?

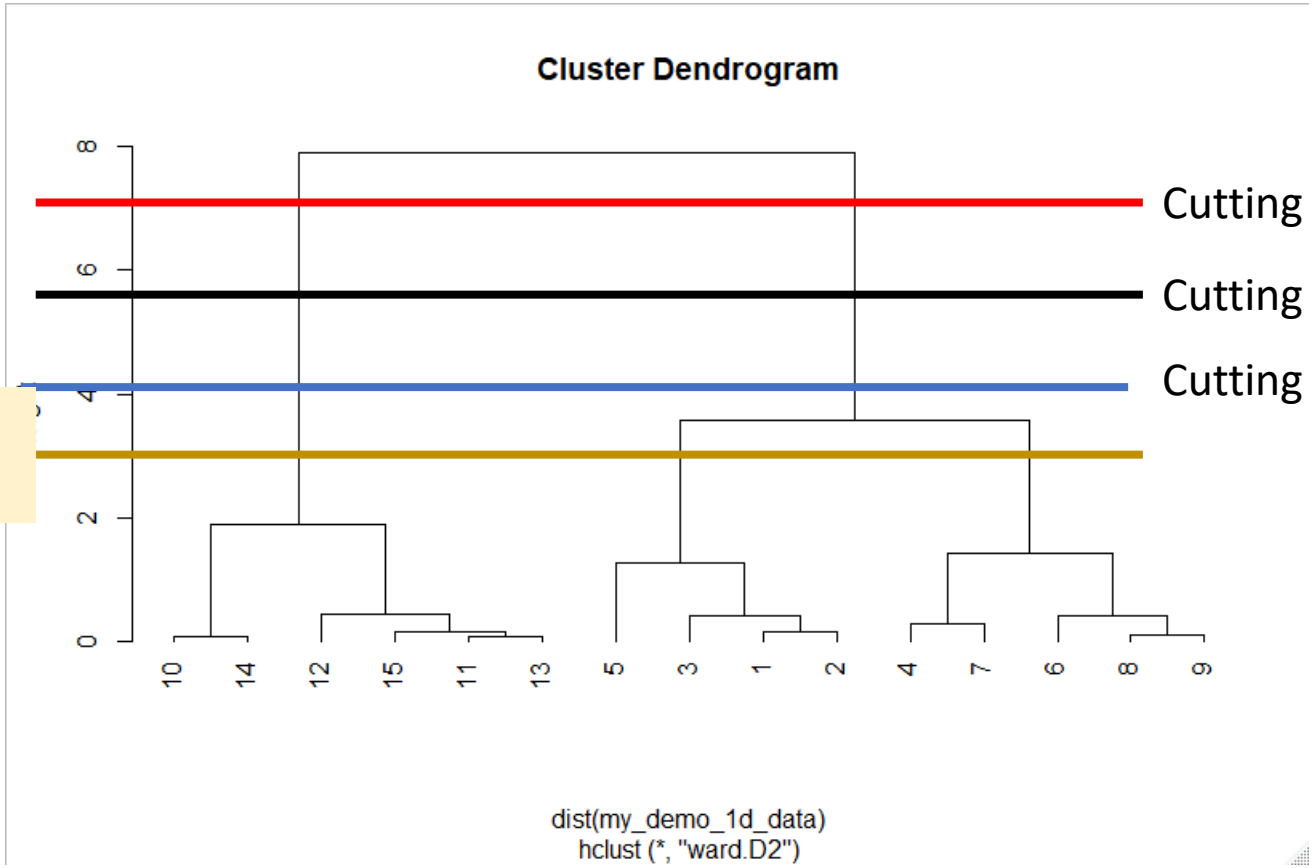


Which height should we cut the tree?



Which height should we cut the tree?

Cutting here produces 3 clusters

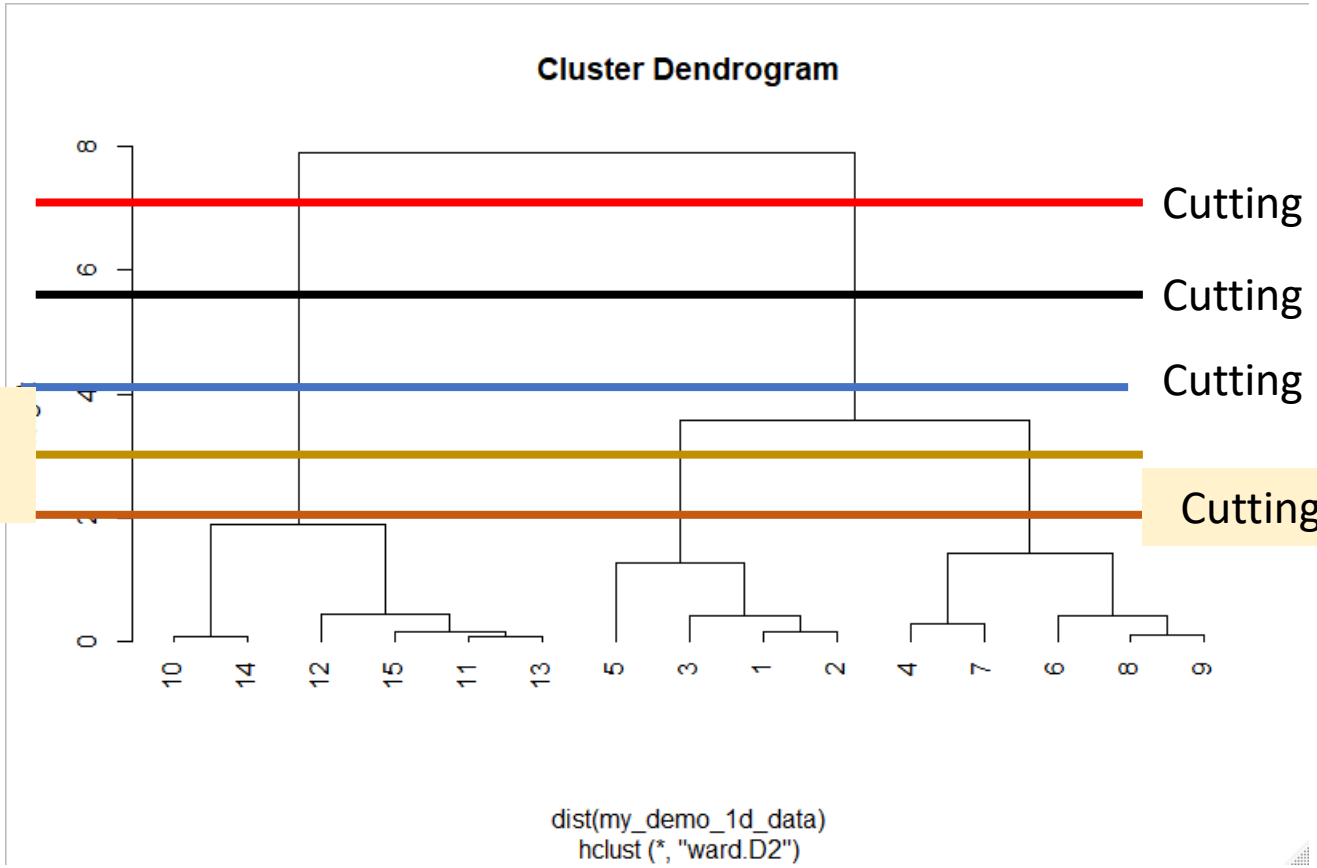


Cutting here produces 2 clusters

Cutting here produces 2 clusters

Cutting here produces 2 clusters

Which height should we cut the tree?



Cutting here produces 2 clusters

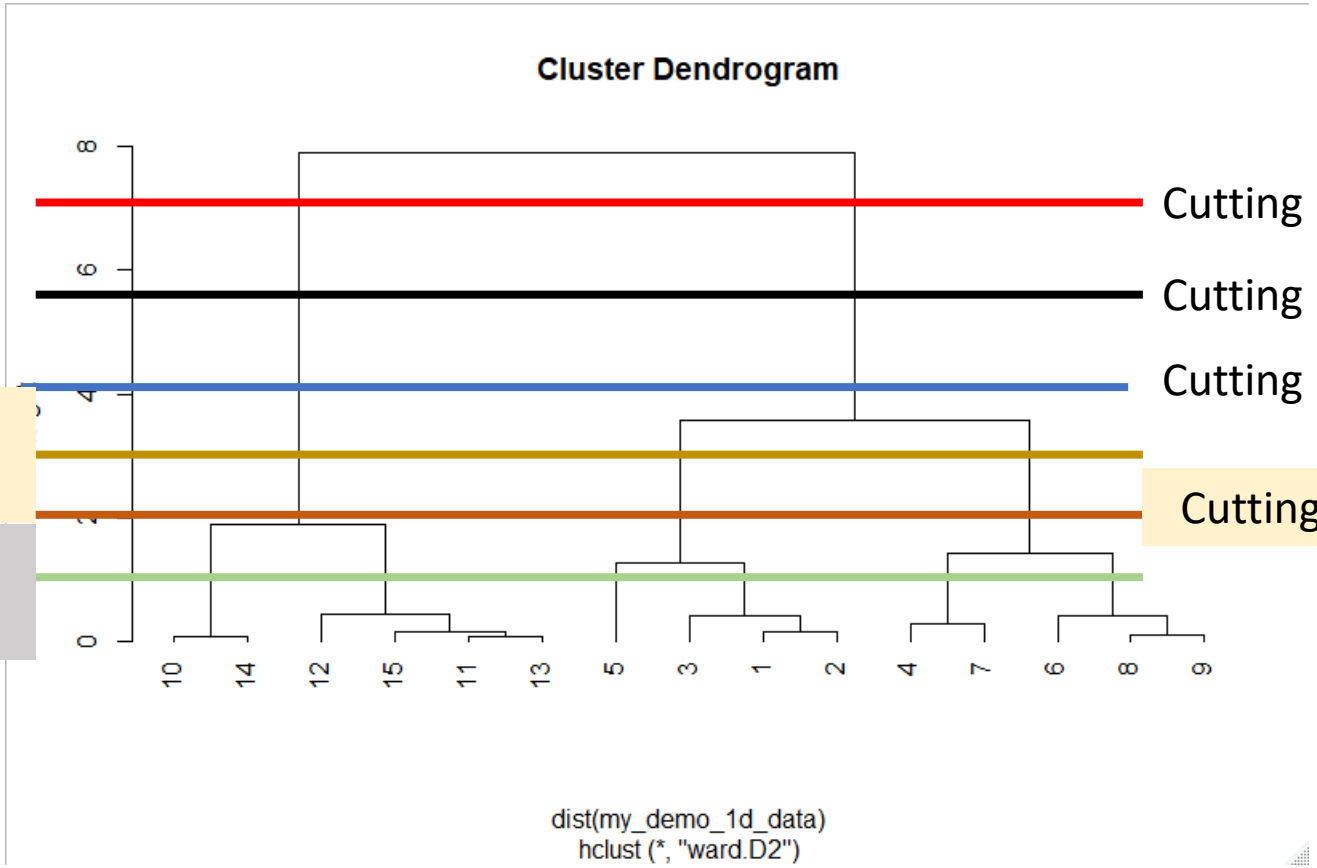
Cutting here produces 2 clusters

Cutting here produces 2 clusters

Cutting here produces 3 clusters

Cutting here produces 3 clusters

Which height should we cut the tree?



Cutting here produces 2 clusters

Cutting here produces 2 clusters

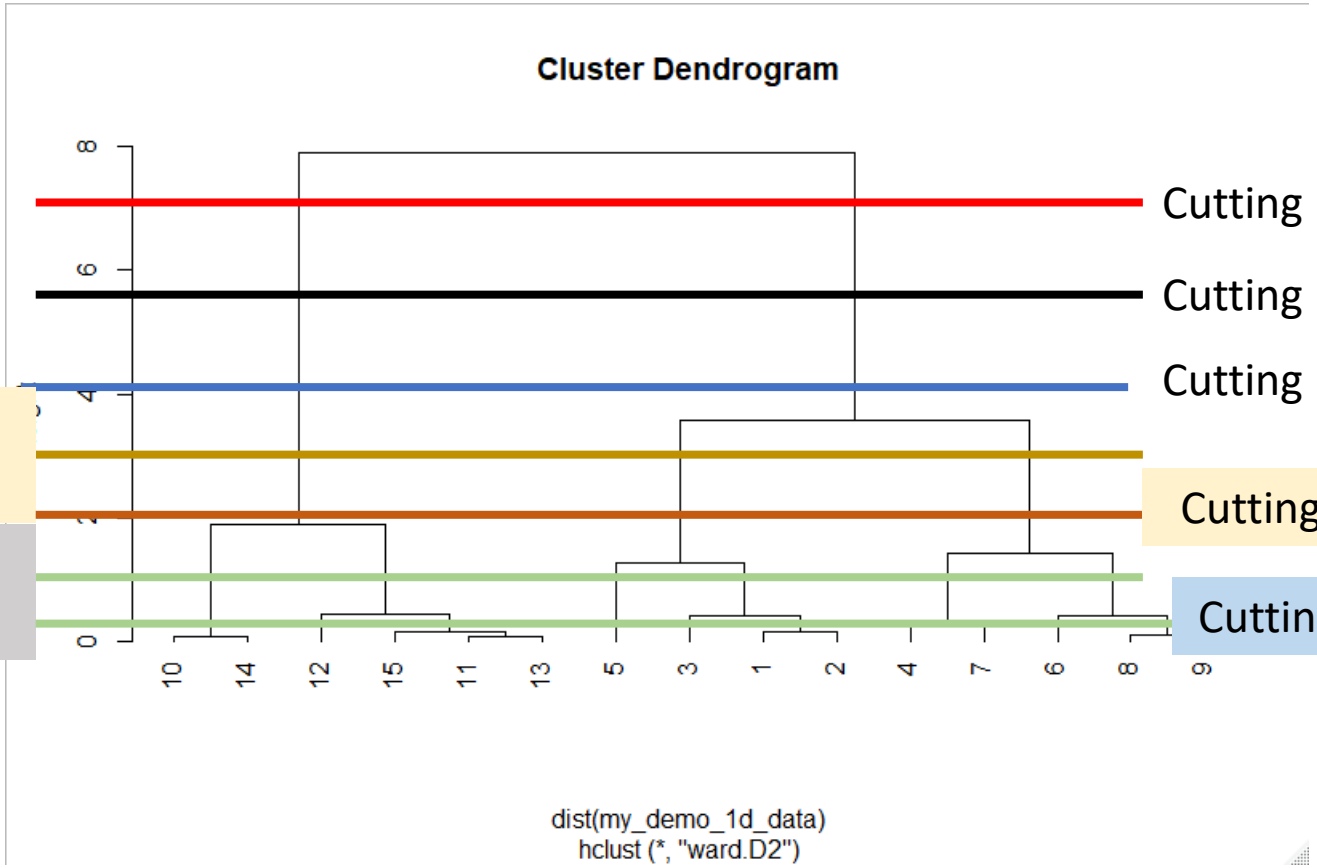
Cutting here produces 2 clusters

Cutting here produces 3 clusters

Cutting here produces 3 clusters

Cutting here produces 6 clusters

Which height should we cut the tree?



Cutting here produces 2 clusters

Cutting here produces 2 clusters

Cutting here produces 2 clusters

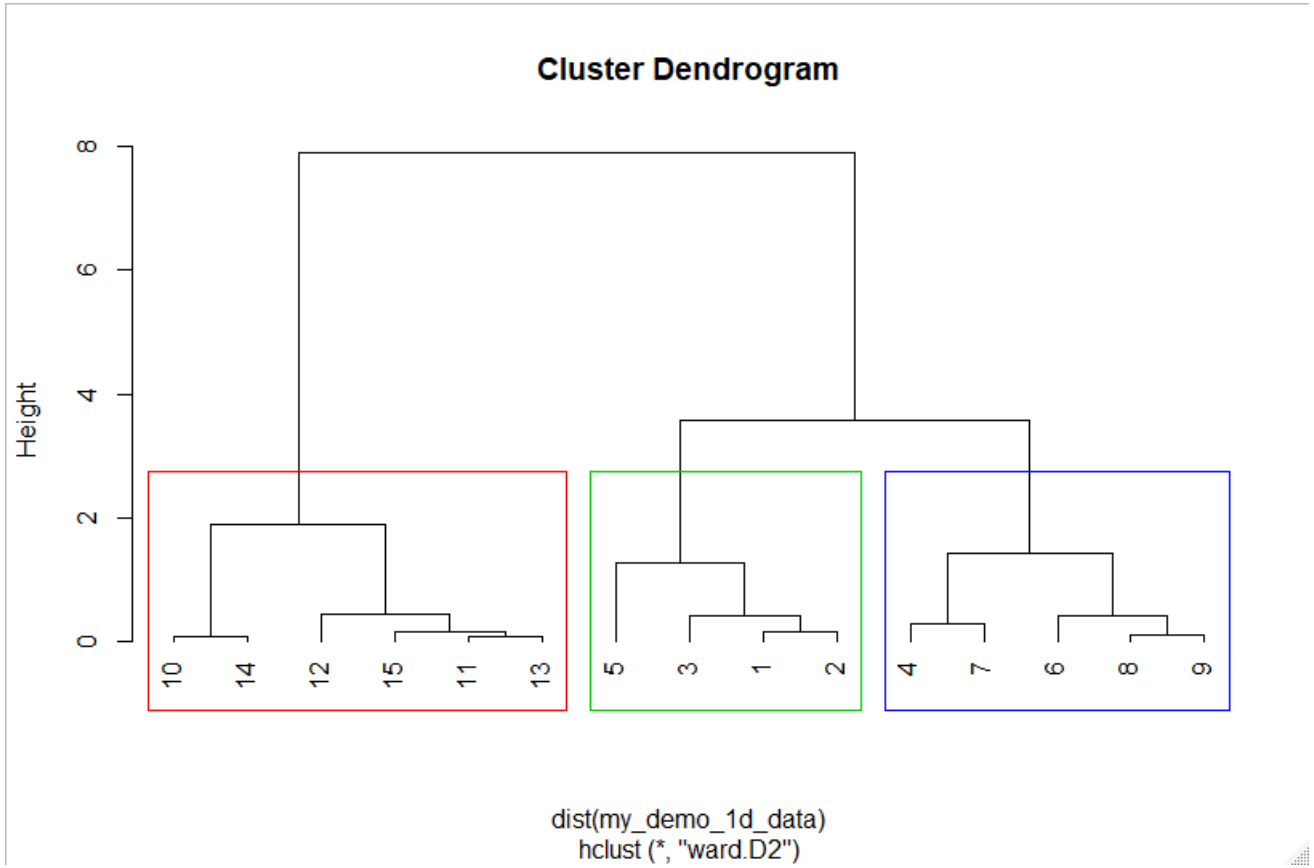
Cutting here produces 3 clusters

Cutting here produces 3 clusters

Cutting here produces 6 clusters

Cutting here produces 10 clusters

Cut to maximize the resulting number of clusters which have minimal sensitivity to the height of the cut.



IMPORTANT: we discussed defining similarity based on the Squared Euclidean distance

- Other distance metrics exist and could be used.
  - Manhattan distance calculates distance based on ABSOLUTE differences instead of SQUARED differences.

- There are also correlation-based distance metrics.

- There are even distance metrics designed to allow for categorical variables!!!!

- The choice of distance metric could change clustering results, especially at high dimensions!!

Hierarchical vs KMeans

- Both rely on DISTANCES.
- Hierarchical ALSO requires specifying the LINKAGE!
- Both require pre-processed columns.
- Hierarchical does NOT require specifying the number of clusters upfront.
- Hierarchical produces a useful visualization to inspect the results – the DENDROGRAM.

Practical concerns with Hierarchical

- Agglomerative hierarchical methods are usually limited to roughly 10000 to 15000 observations.
  - Due to calculating and storing the distances between ALL pairs of observations.

- Therefore, if you are working with MORE than 15000 observations, you should produce a RANDOM subset of data to start using hierarchical clustering.

- Less issues with the number of rows with KMeans. You can apply KMeans to more than 20000 rows without any concern.