Clustering

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

K-MEANS

1. Select a number of clusters K.
2. Let $C_1, \ldots, C_K$ partition $\{1, 2, 3, \ldots, n\}$ such that
   - All observations belong to some set C_j.
   - No observation belongs to more than one set.
3. K-means attempts to form these sets by making within-cluster variation, $W(C_k)$, as small as possible.

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k).$$

4. To Define W, we need a concept of distance. By far the most common is Euclidean

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

That is, the average (Euclidean) distance between all cluster members.

K-means

It turns out

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k). \tag{1}$$

is too hard of a problem to solve computationally (K^n partitions!).

So, we make a greedy approximation:

1. Randomly assign observations to the K clusters
2. Iterate until the cluster assignments stop changing:
   - For each of K clusters, compute the centroid, which is the p-length vector of the means in that cluster.
   - Assign each observation to the cluster whose centroid is closest (in Euclidean distance).

This procedure is guaranteed to decrease (1) at each step.

K-means: A Summary

To fit K-means, you need to

1. Pick K (inherent in the method)
2. Convince yourself you have found a good solution (due to the randomized approach to the algorithm).

It turns out that 1. is difficult to do in a principled way. We will discuss this next

For 2., a commonly used approach is to run K-means many times with different starting points. Pick the solution that has the smallest value for

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

Choosing K

Choosing the Number of Clusters

Why is it important?

- It might make a big difference (concluding there are $K = 2$ cancer sub-types versus $K = 3$).
- One of the major goals of statistical learning is automatic inference. A good way of choosing K is certainly a part of this.

Reminder: What does K-means do?

Given a number of clusters K, we (approximately) minimize:

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

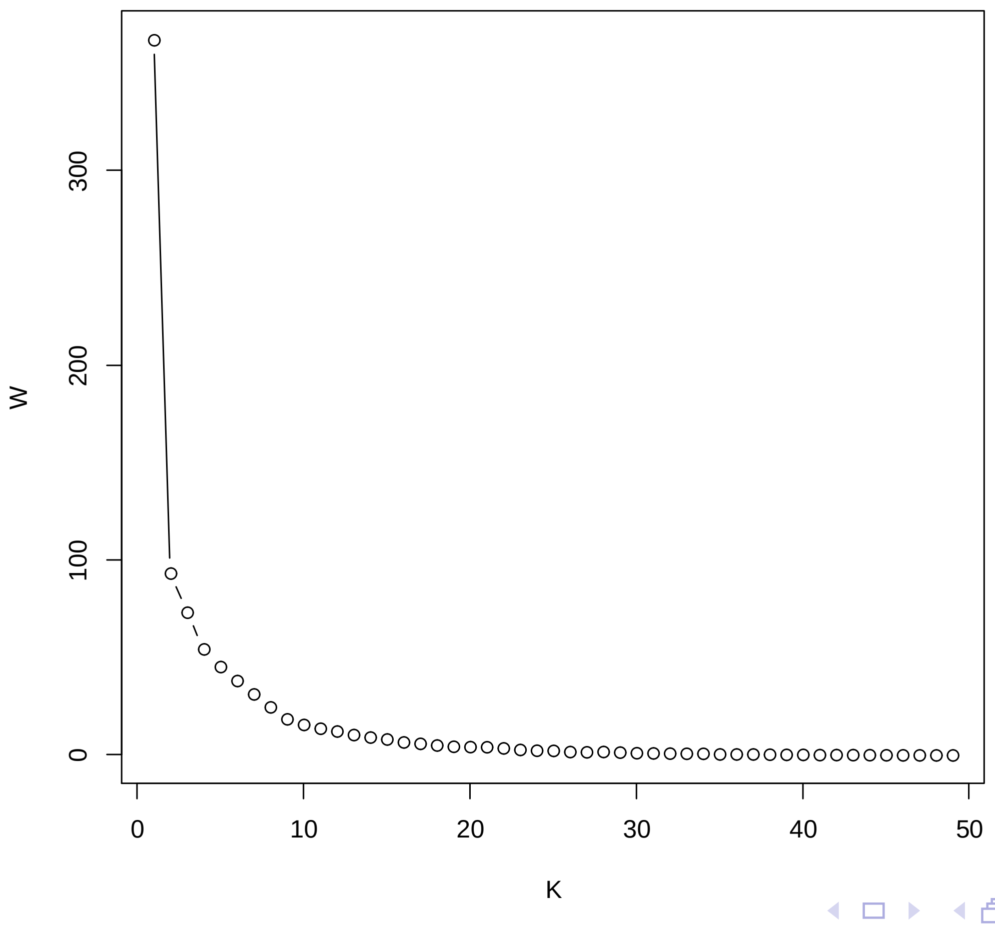
We can rewrite this in terms of the centroids as

$$W(K) = \sum_{k=1}^{K} \sum_{i \in C_k} ||X_i - \overline{X}_k||_2^2,$$

Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

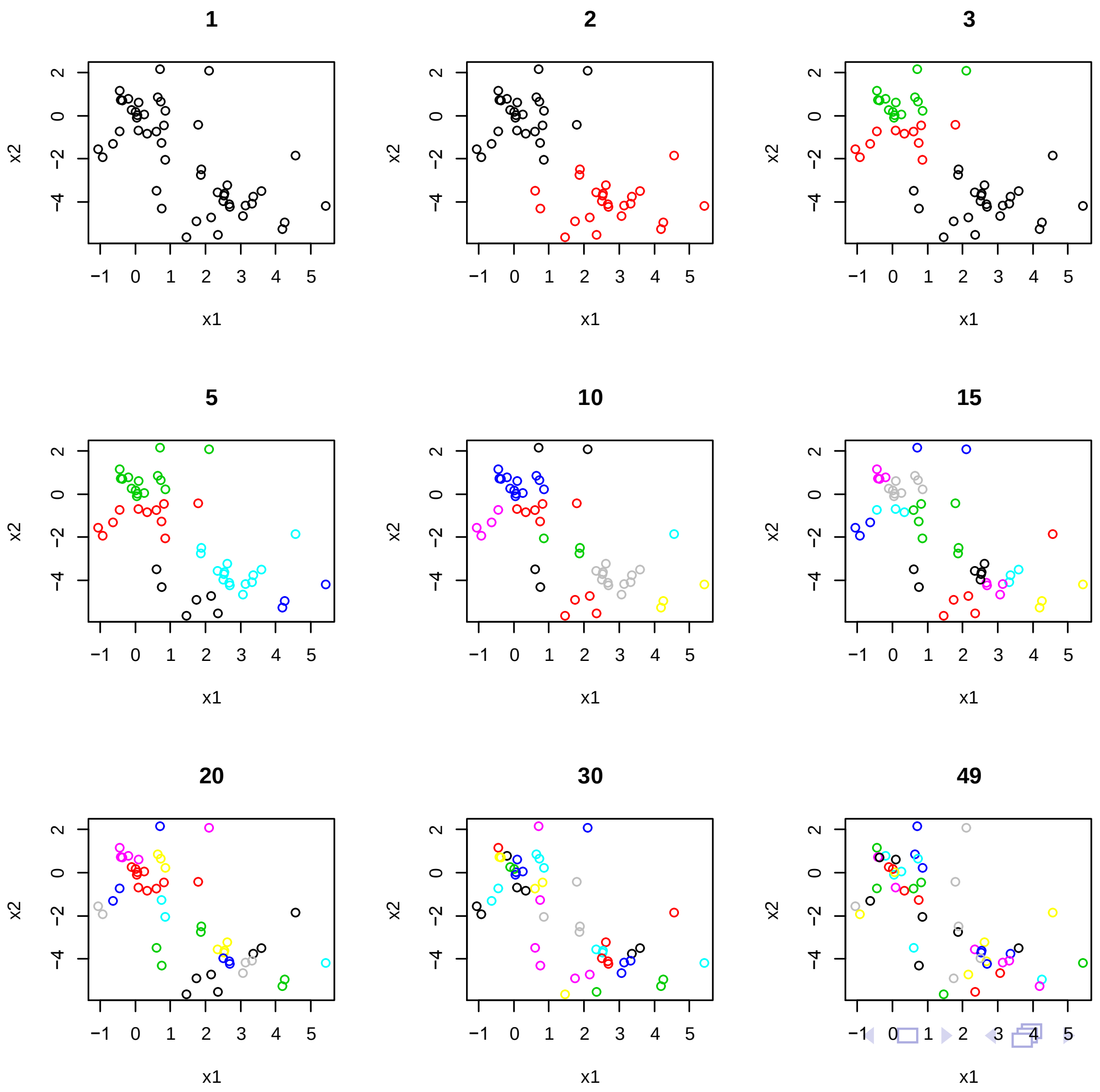
```
plotW = rep(0,49)
for(K in 1:49){
  plotW[K] =  kmeans(x,centers=K,nstart=20)$tot.withinss
}
```



Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

A look at the cluster solution



Between-cluster variation

Within-cluster variation measures how tightly grouped the clusters are. As we increase K, this will always decrease.

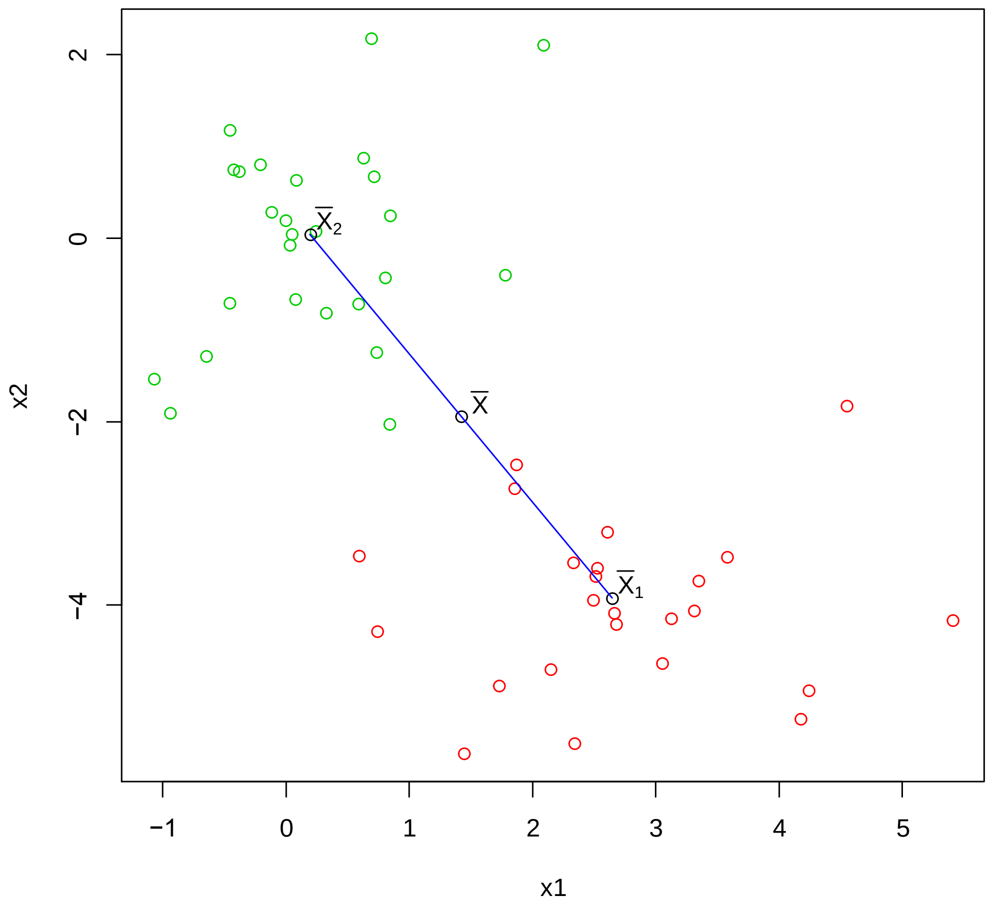
What we are missing is between-cluster variation, ie: how spread apart the groups are

$$B = \sum_{k=1}^{K} |C_k| ||\overline{X}_k - \overline{X}||_2^2,$$

where $|C_k|$ is the number of points in C_k, and $\overline{X}$ is the grand mean of all observations:

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

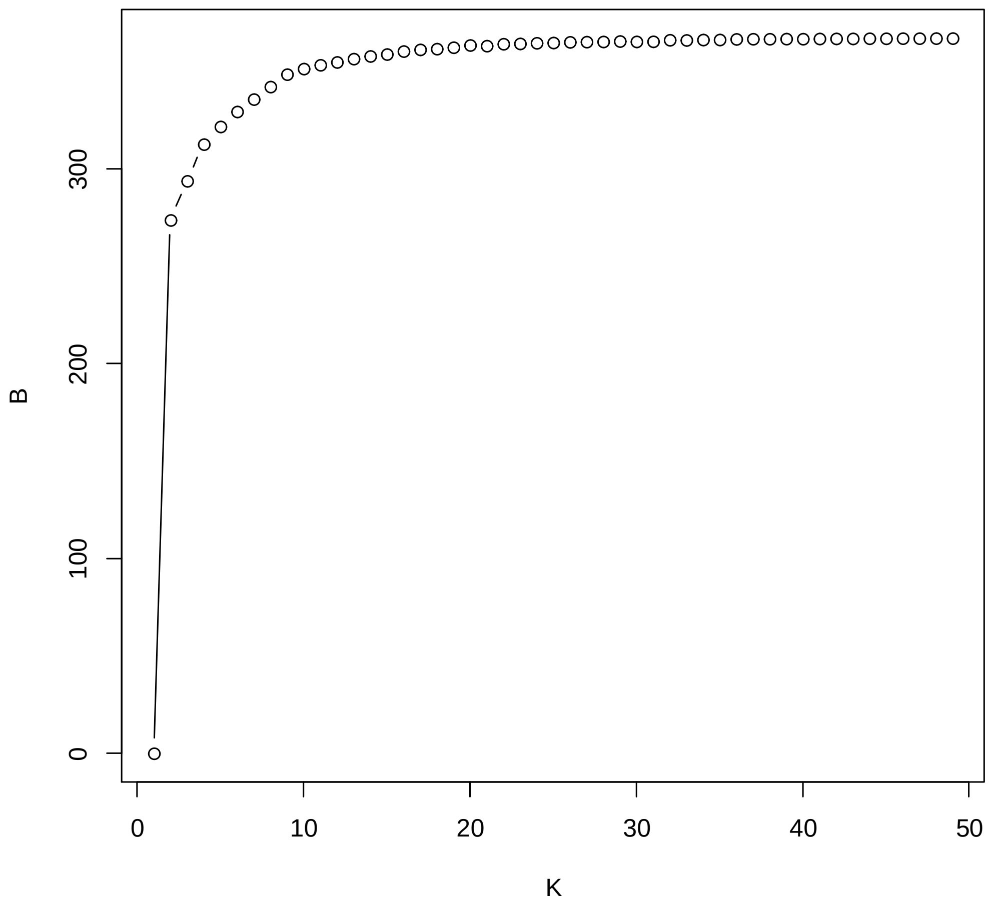
Between-cluster variation: Example



$$B = |C_1| \| \overline{X}_1 - \overline{X} \|_2^2 + |C_2| \| \overline{X}_2 - \overline{X} \|_2^2$$
$$W = \sum_{i \in C_1} |C_1| \| \overline{X}_1 - X_i \|_2^2 + \sum_{i \in C_2} |C_2| \| \overline{X}_2 - X_i \|_2^2$$

Can we just maximize B?

Sadly, no. Just like W can be made arbitrarily small, B will always be increasing with increasing K.



CH index

Ideally, we would like our cluster assignment to simultaneously have small W and large B.

This is the idea behind CH index. For clustering assignments coming from K clusters, we record CH score:

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

To choose K, pick some maximum number of clusters to be considered ($K_{\max} = 20$, for example) and choose the value of K that

$$\hat{K} = \underset{K \in \{2,\dots,K_{\max}\}}{\arg\max} \; CH(K).$$

Note: CH is undefined for $K = 1$.

(Calinski, Harabasz (1974))

CH INDEX

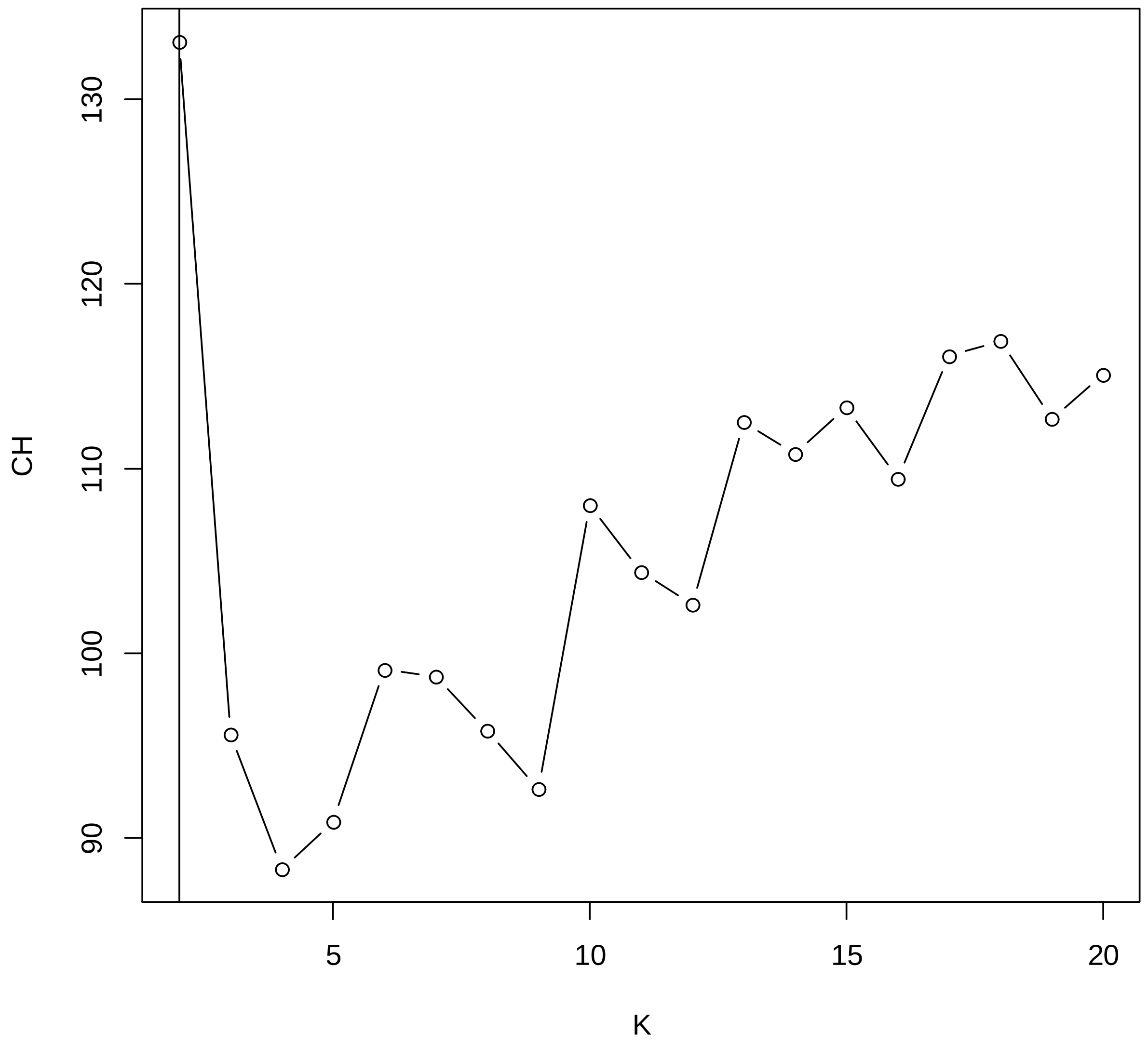
```
ch.index = function(x,kmax,iter.max=100,nstart=10,
                      algorithm="Lloyd")
{
  ch = numeric(length=kmax-1)
  n = nrow(x)
  for (k in 2:kmax) {
    a = kmeans(x,k,iter.max=iter.max,nstart=nstart,
              algorithm=algorithm)
    w = a$tot.withinss
    b = a$betweenss
   ch[k-1] = (b/(k-1))/(w/(n-k))
  }
  return(list(k=2:kmax,ch=ch))
}
```

A simulated example

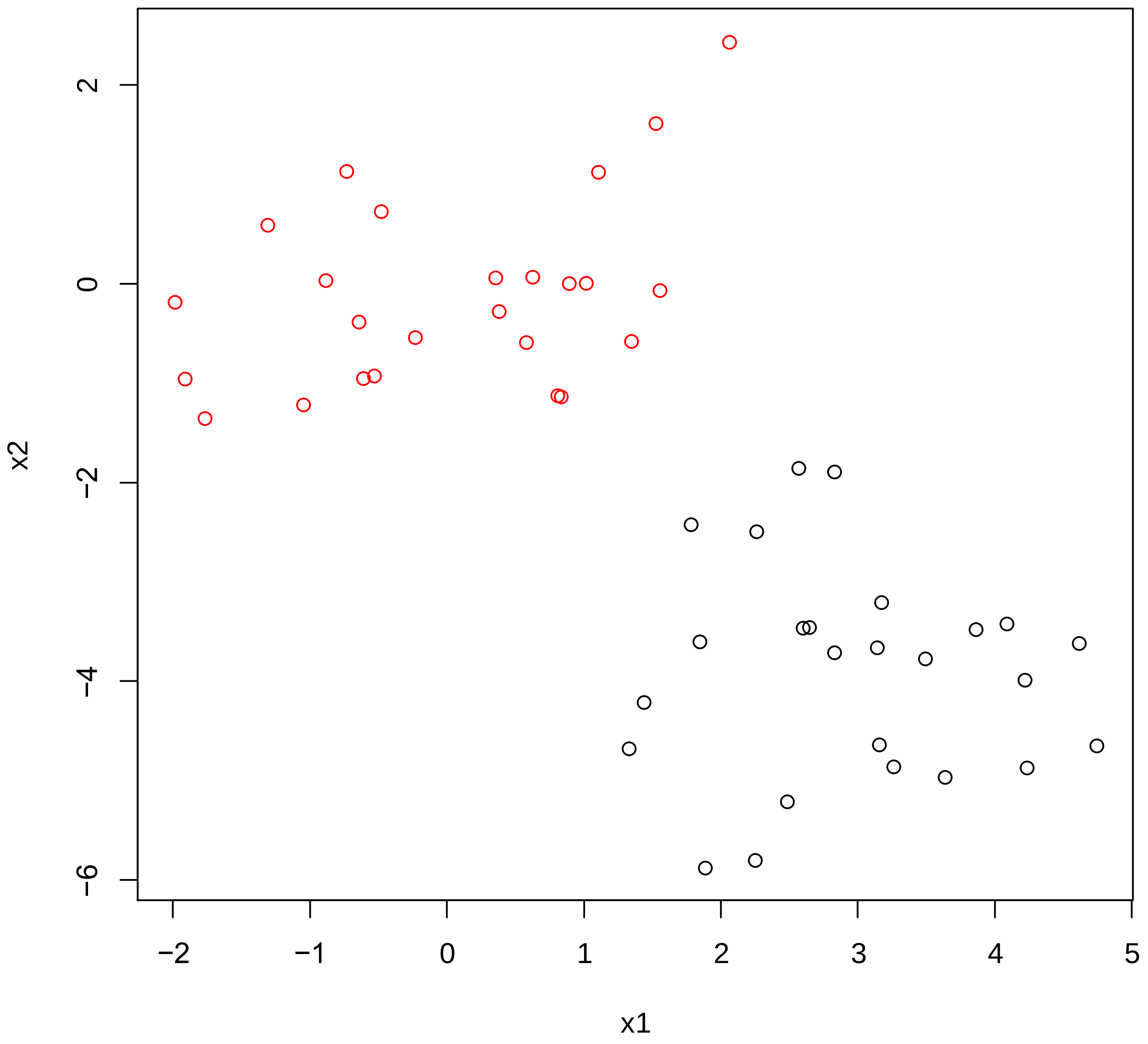
```
x = matrix(rnorm(50*2),ncol=2)
x[1:25,1] = x[1:25,1] + 3
x[1:25,2] = x[1:25,2] -4
```

We want to cluster this data set using K-means with K chosen via CH index.

CH plot



Corresponding solution



Hierarchical clustering

From K-means to hierarchical clustering

Recall two properties of K-means clustering

1. It fits exactly K clusters.
2. Final clustering assignments depend on the chosen initial cluster centers.

Alternatively, we can use hierarchical clustering. This has the advantage that

1. No need to choose the number of clusters before hand.
2. There is no random component (nor choice of starting point).

From K-means to hierarchical clustering

There is a catch: we need to choose a way to measure the distance between clusters, called the linkage.

Given the linkage, hierarchical clustering produces a sequence of clustering assignments.

At one end, all points are in their own cluster.

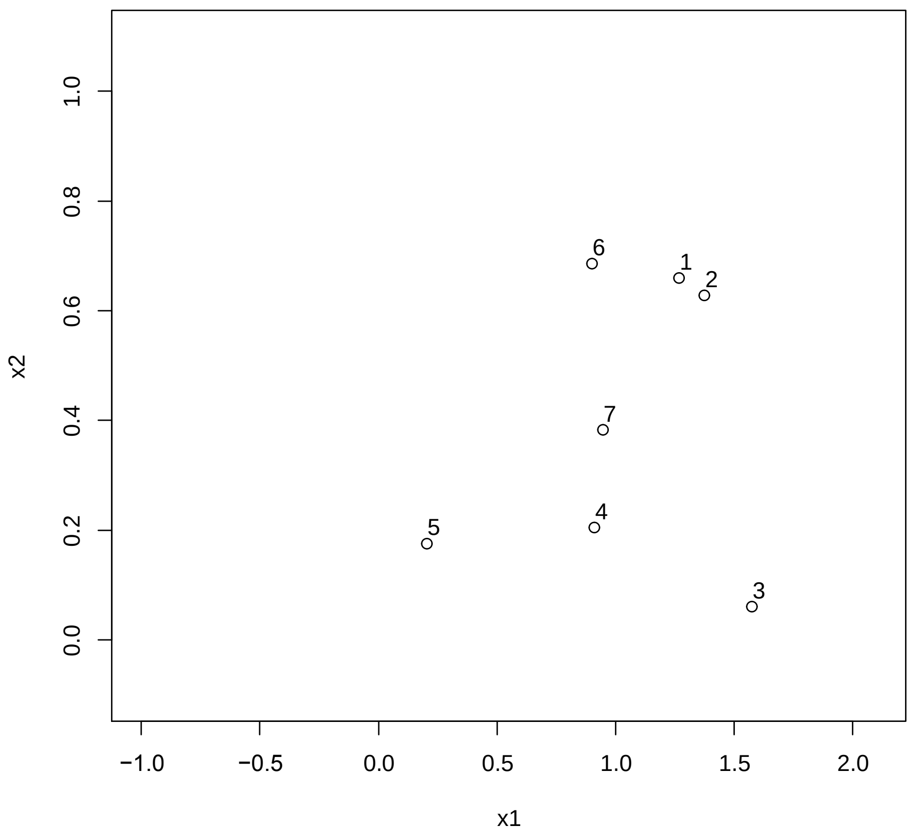
At the other, all points are in one cluster.

In the middle, there are nontrivial solutions.

Agglomerative example

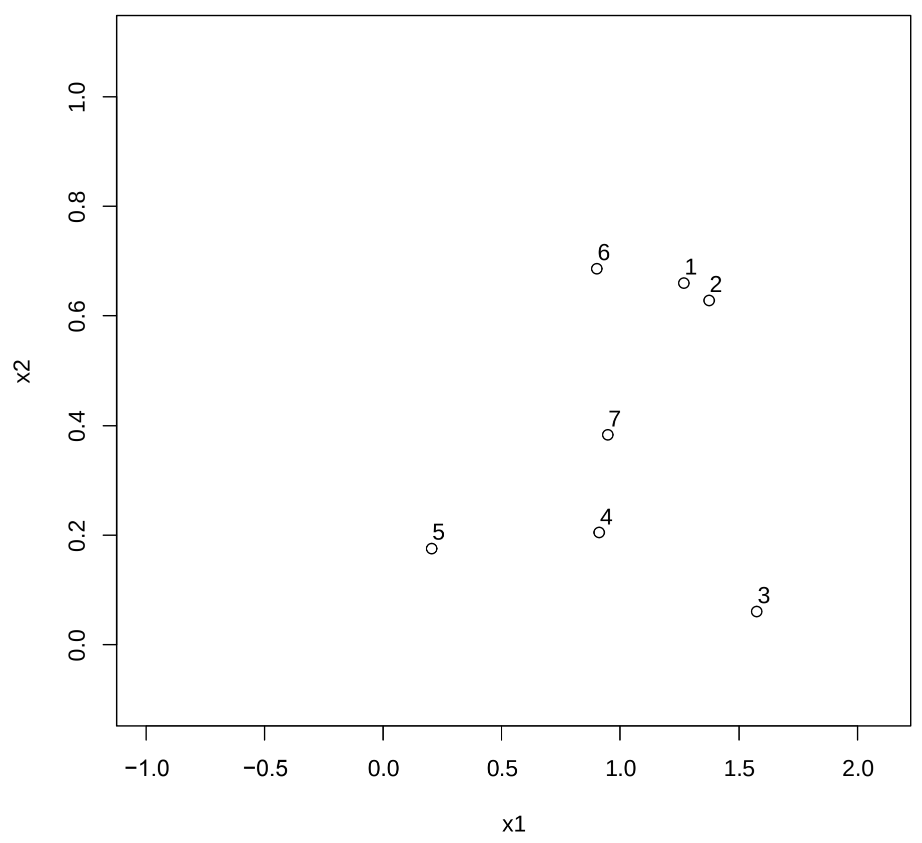
Given these data points, an agglomerative algorithm might decide on the following clustering sequence:

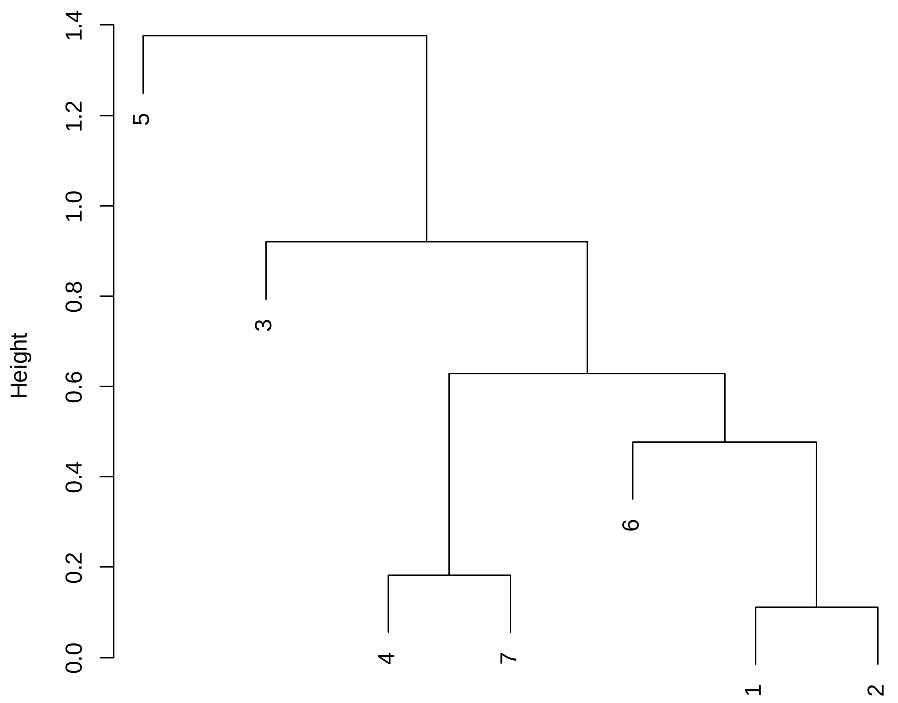
(Important: Different choices of linkage would result in different solutions)



1. $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$
2. $\{1, 2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$
3. $\{1, 2\}, \{3\}, \{5\}, \{4, 7\}$
4. $\{1, 2, 6\}, \{3\}, \{5\}, \{4, 7\}$
5. $\{1, 2, 4, 6, 7\}, \{3\}, \{5\}$
6. $\{1, 2, 3, 4, 6, 7\}, \{5\}$
7. $\{1, 2, 3, 4, 5, 6, 7\}$

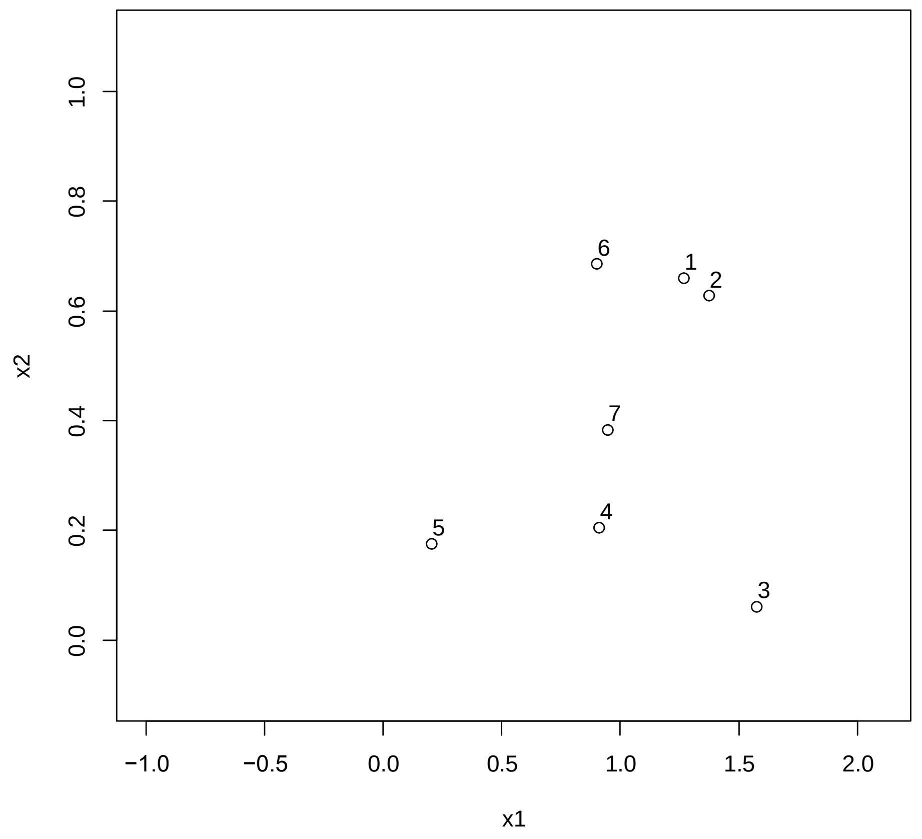
We can also represent the sequence of clustering assignments as a dendrogram

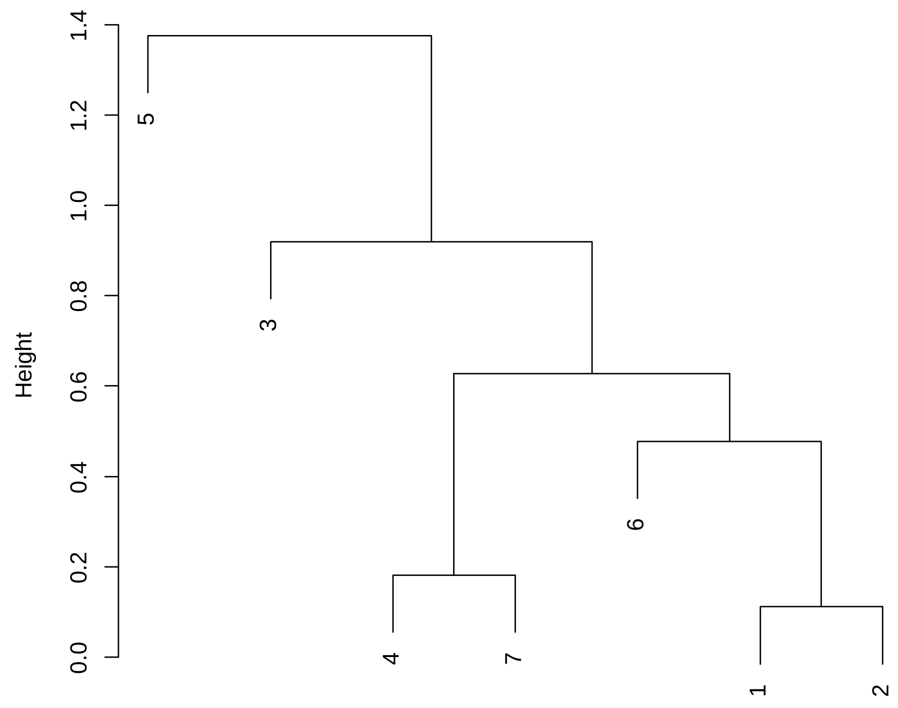




Note that cutting the dendrogram horizontally partitions the data points into clusters

Back to the example





For instance, the linkage distance between the cluster $\{4, 7\}$ and the cluster $\{1, 2, 6\}$ is about .65.

Linkages

Notation: Define $X_1, \ldots, X_n$ to be the data

Let the dissimiliarities be d_{ij} between each pair X_i, X_j

At any level, clustering assignments can be expressed by sets $G = \{i_1, i_2, \ldots, i_r\}$. given the indices of points in this group. Define $|G|$ to be the size of G.

Linkage: The function $d(G, H)$ that takes two groups G, H and returns the linkage distance between them.

Agglomerative clustering, given the linkage:

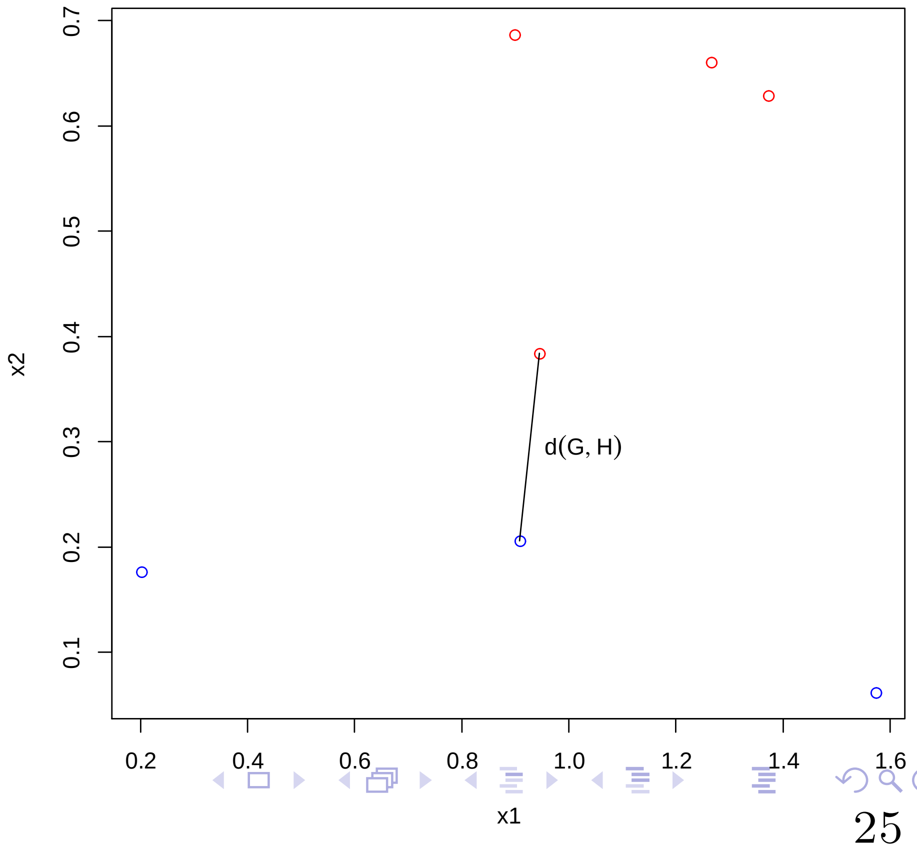
- Start with each point in its own group
- Until there is only one cluster, repeatedly merge the two groups G, H that minimize $d(G, H)$.

Single linkage

In single linkage (a.k.a nearest-neighbor linkage), the linkage distance between G, H is the smallest dissimilarity between two points in different groups:

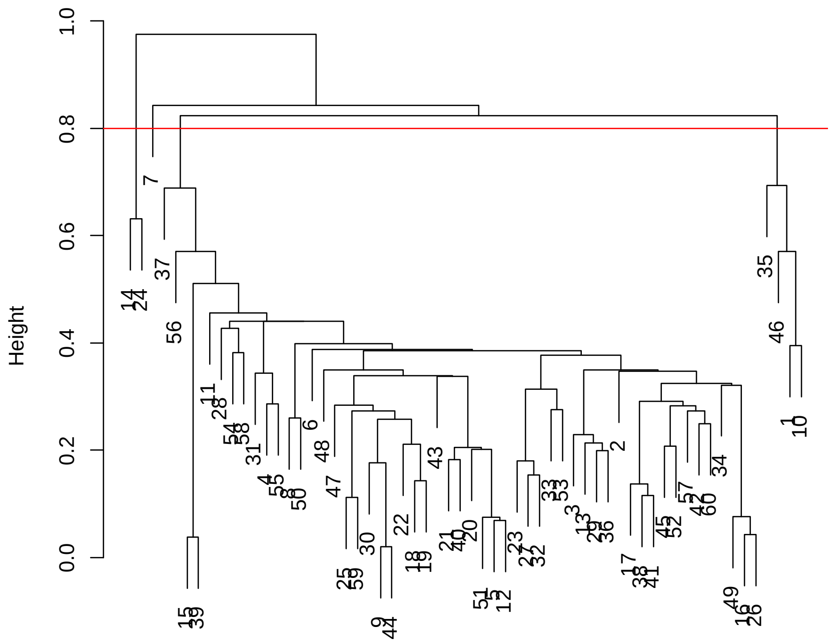
$$d_{\text{single}}(G, H) = \min_{i \in G, j \in H} d_{ij}$$

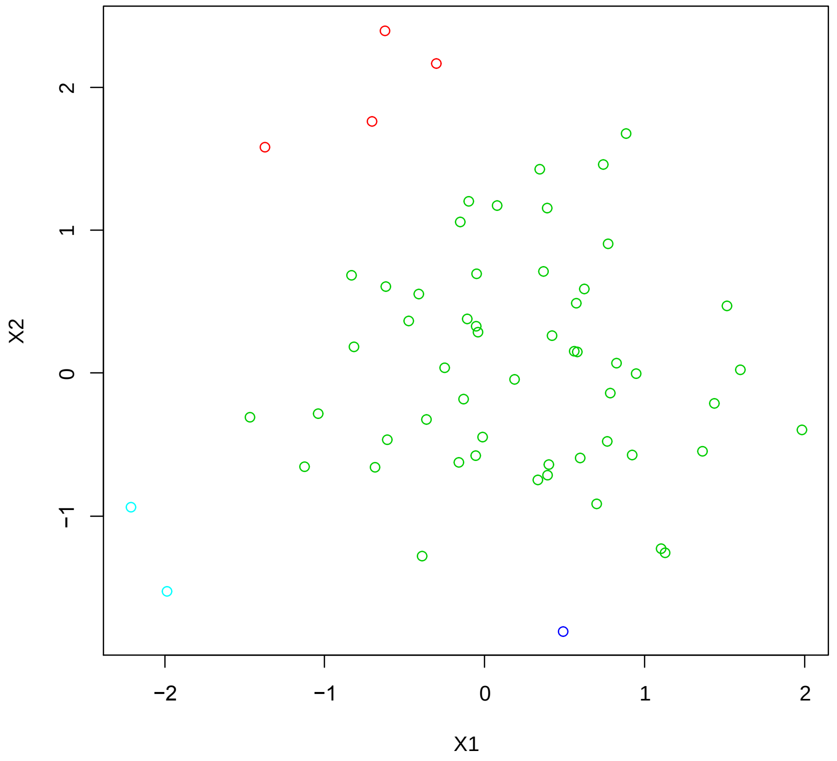
Example: There are two clusters G and H (red and blue).
The single linkage score
(i.e. $d_{\text{single}}(G, H)$)
is the dissimilarity between the closest pair
(length of black line segment)



Single linkage example

Here $n = 60$, $X_i \in \mathbb{R}^2$, $d_{ij} = ||X_i - X_j||_2$. Cutting the tree at $h = 0.8$ gives the cluster assignments marked by colors





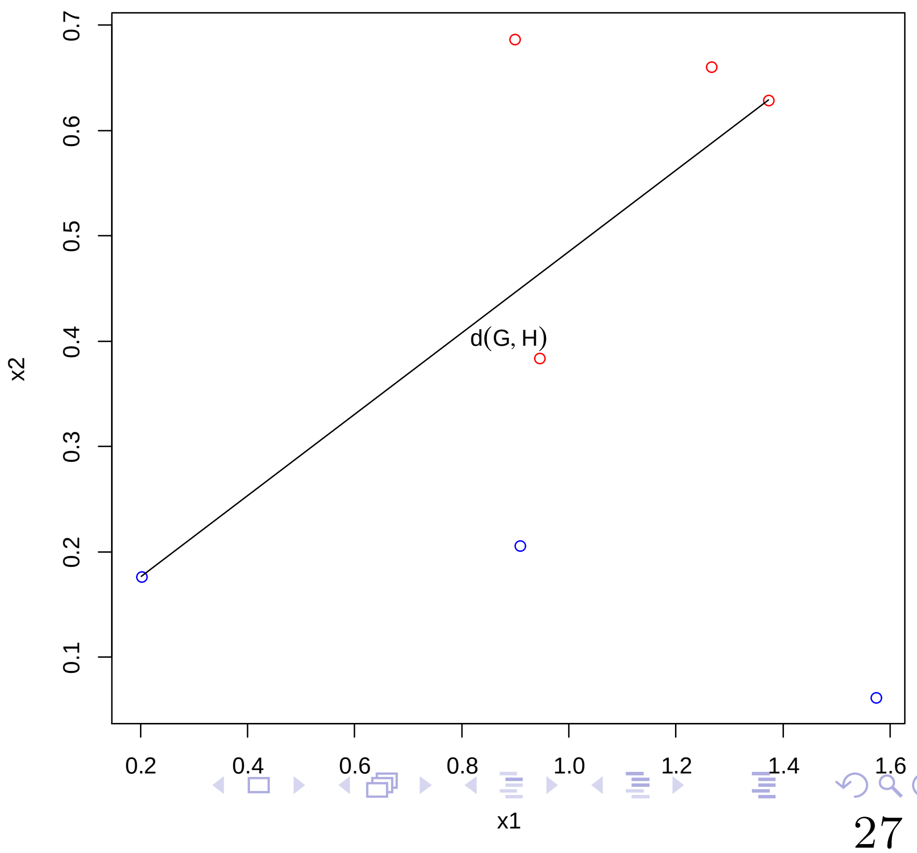
Cut interpretation: For each point X_i, there is another point X_j in the same cluster with $d_{ij} \le 0.8$ (assuming more than 1 point in cluster). Also, no points in different clusters are closer than 0.8.

Complete linkage

In complete linkage (i.e. farthest-neighbor linkage), linkage distance between G, H is the largest dissimilarity between two points in different clusters:

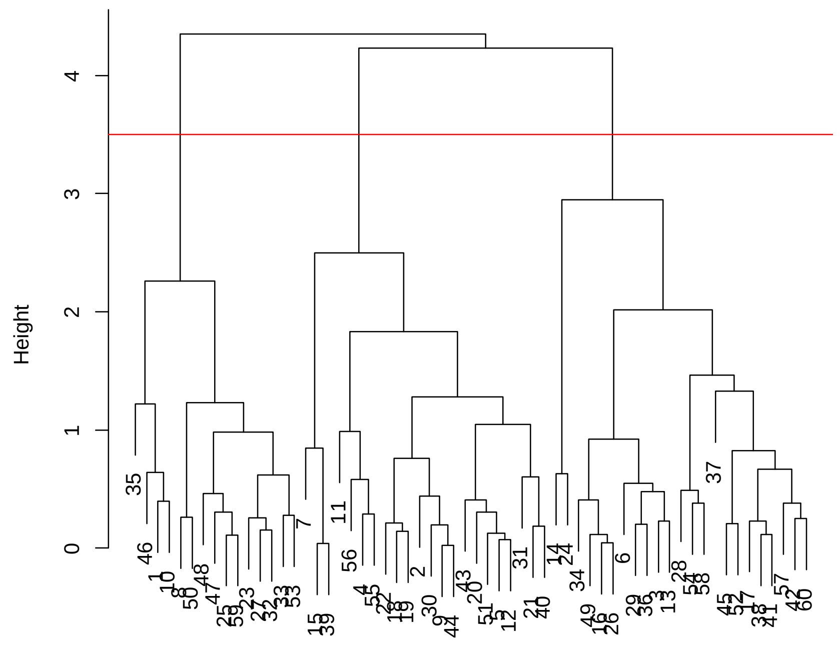
$$d_{\text{complete}}(G, H) = \max_{i \in G, j \in H} d_{ij}.$$

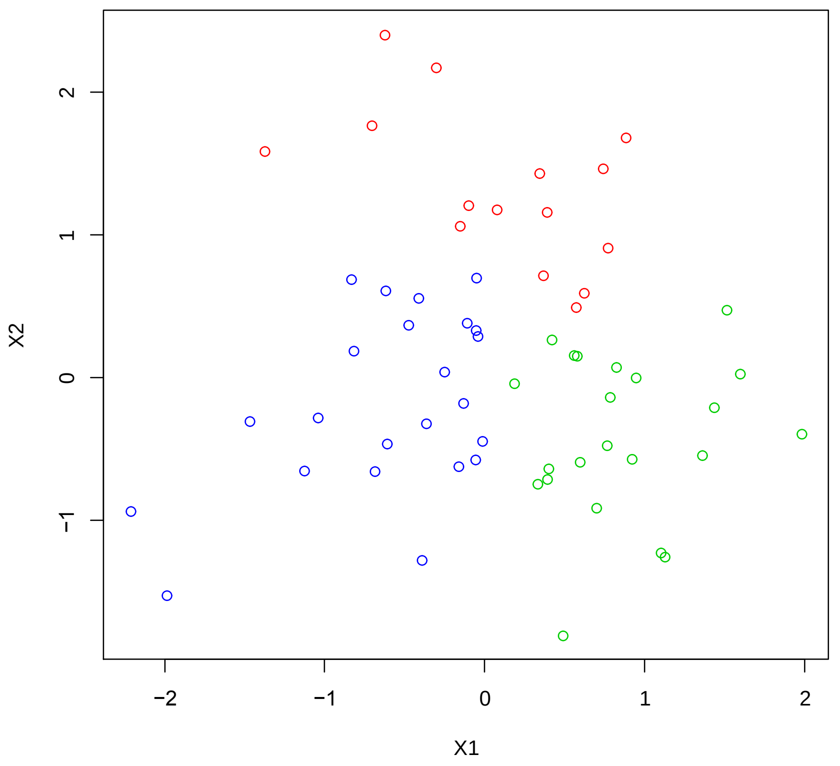
Example: There are two clusters G and H (red and blue).
The complete linkage score
(i.e. $d_{\text{complete}}(G, H)$)
is the dissimilarity between the farthest pair
(length of black line segment)



Complete linkage example

Same data as before. Cutting the tree at $h = 3.5$ gives the clustering assignment





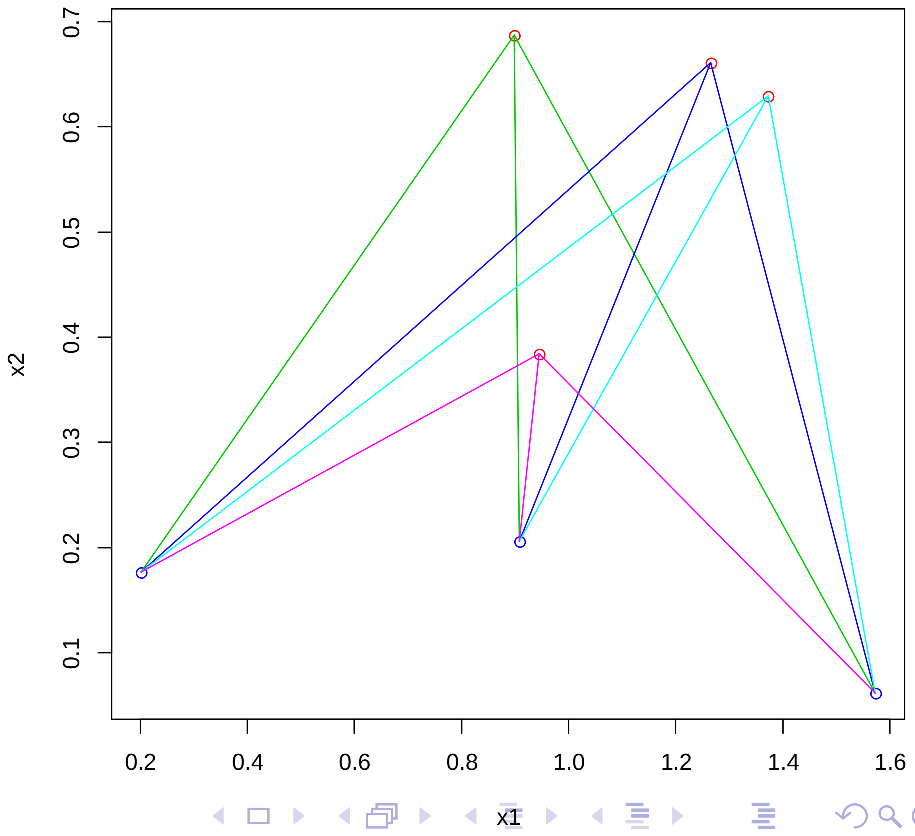
Cut interpretation: For each point X_i, every other point X_j in the same cluster has $d_{ij} \leq 3.5$.

Average linkage

In average linkage, the linkage distance between G, H is the average dissimilarity over all points in different clusters:

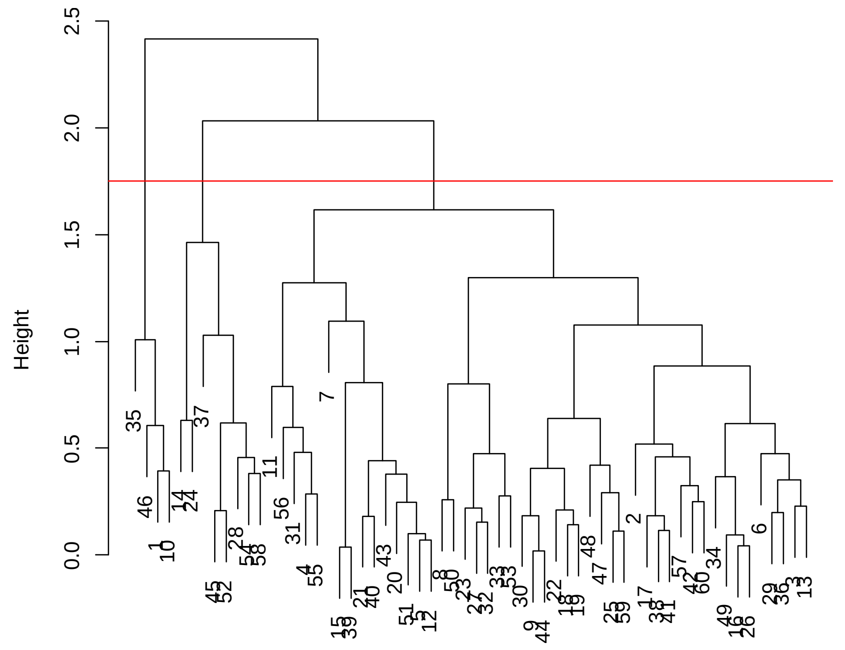
$$d_{\text{average}}(G, H) = \frac{1}{|G| \cdot |H|} \sum_{i \in G, j \in H} d_{ij}.$$

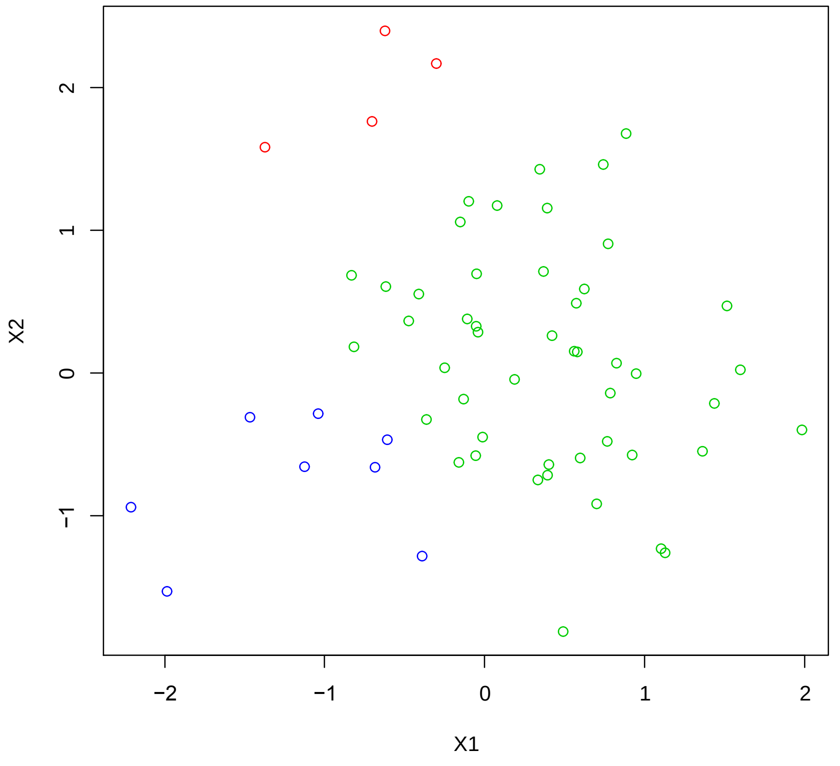
Example: There are two clusters G and H (red and blue).
The average linkage score
(i.e. $d_{\text{average}}(G, H)$)
is the average dissimilarity between all points in different clusters
(average of lengths of colored line segments)



Average linkage example

Same data as before. Cutting the tree at $h = 1.75$ gives the clustering assignment





Cut interpretation: ??

Common properties

Single, complete, and average linkage share the following:

- They all operate on the dissimilarities d_{ij}. This means that the points we are clustering can be quite general (number of mutations on a genome, polygons, faces, whatever).
- Running agglomerative clustering with any of these linkages produces a dendrogram with no inversions.

No inversions means that the linkage distance between merged clusters only increases as we run the algorithm.

In other words, we can draw a proper dendrogram, where the height of a parent is always higher than the height of either daughter.

(We'll return to this again shortly)

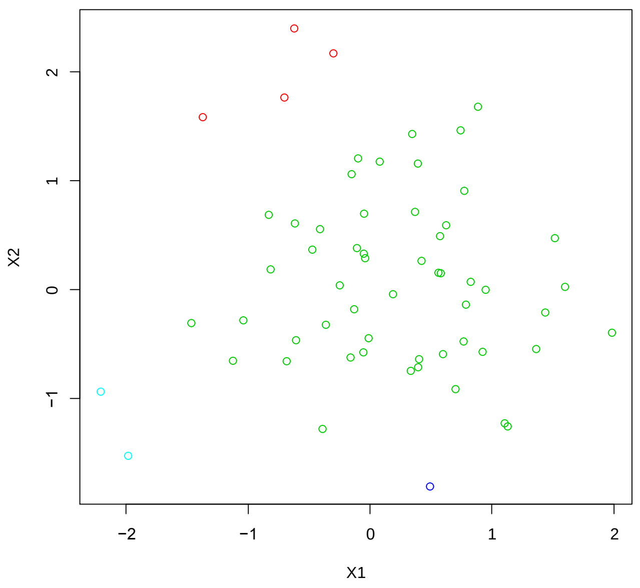
Shortcomings of single and complete linkage

Single and complete linkage have practical problems:

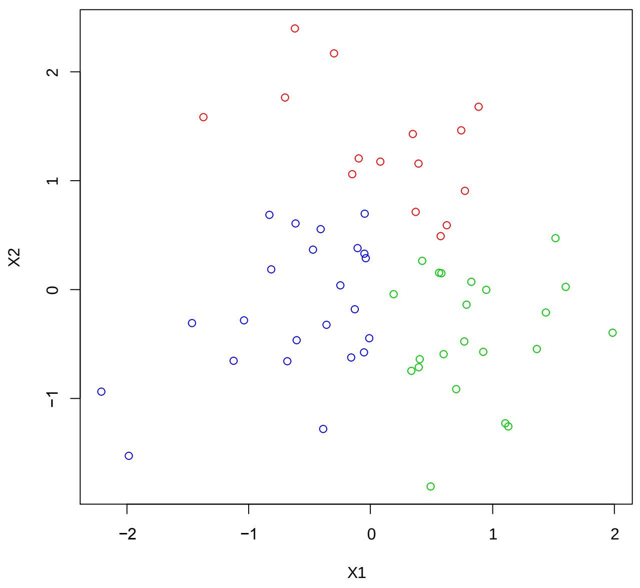
Single linkage:	Often suffers from chaining, that is, we only need a single pair of points to be close to merge two clusters. Therefore, clusters can be too spread out and not compact enough.
Complete linkage:	Often suffers from crowding, that is, a point can be closer to points in other clusters than to points in its own cluster. Therefore, the clusters are compact, but not far enough apart.

Average linkage tries to strike a balance between these two.

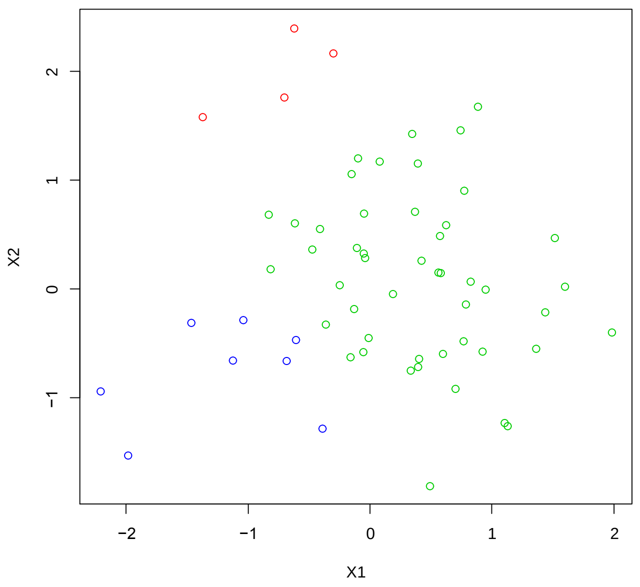
Example of chaining and crowding



Single linkage



Complete linkage



Average linkage

Shortcomings of average linkage

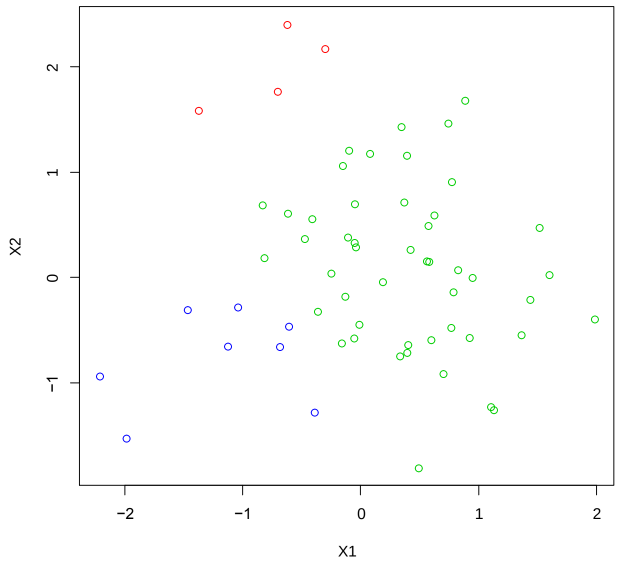
Average linkage isn't perfect.

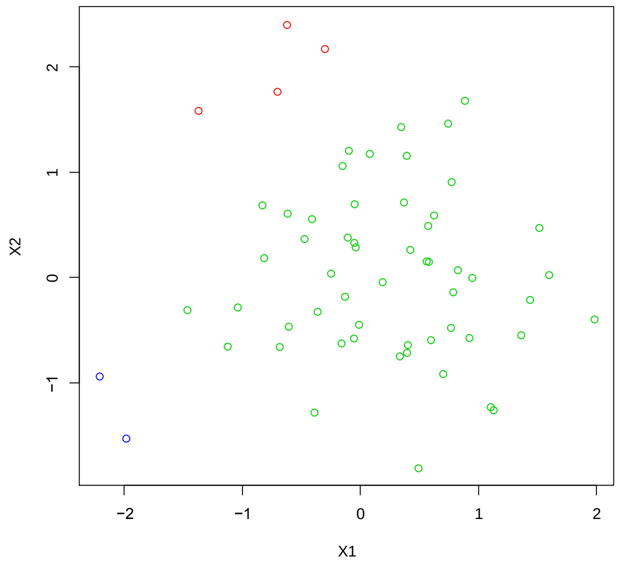
- It isn't clear what properties the resulting clusters have when we cut an average linkage tree.
- Results of average linkage clustering can change with a monotone increasing transformation of the dissimilarities (that is, if we changed the distance, but maintained the ranking of the distances, the cluster solution could change).

Neither of these problems afflict single or complete linkage.

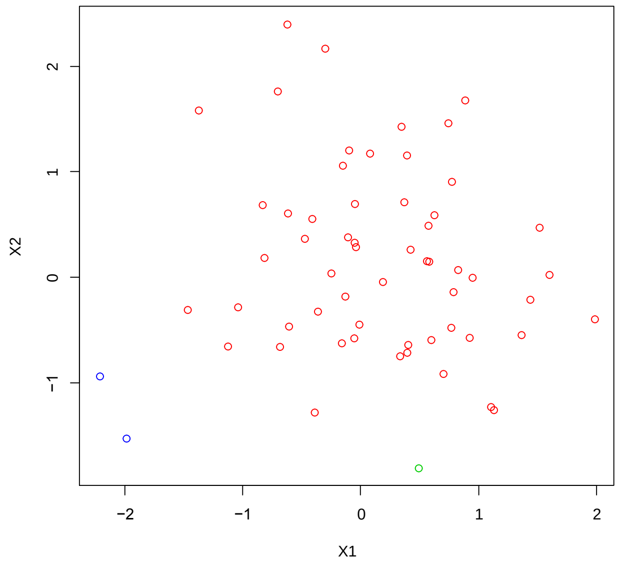
Example of monotone increasing problem

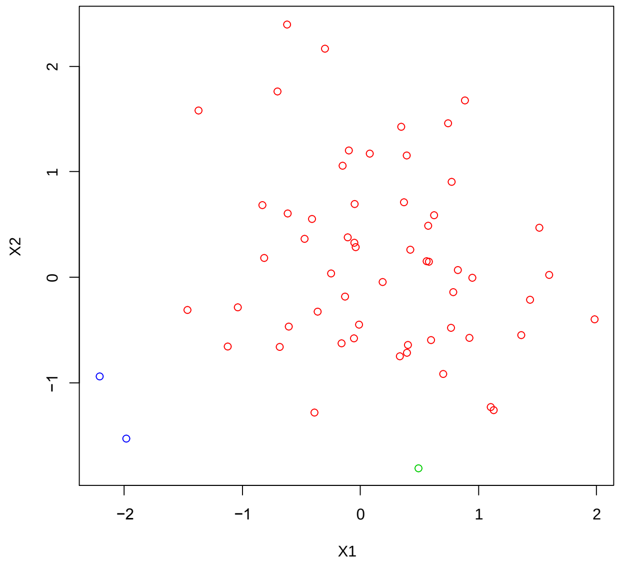
Average





Single





Left: $d_{ij} = ||X_i - X_j||_2$ Right: $d_{ij} = ||X_i - X_j||_2^2$

Hierarchical agglomerative clustering in R

The function `hclust` in base R performs the necessary computations. E.g.

```
Delta = dist(x)
out.average = hclust(Delta,method='average')
plot(out.average)
```

Recap

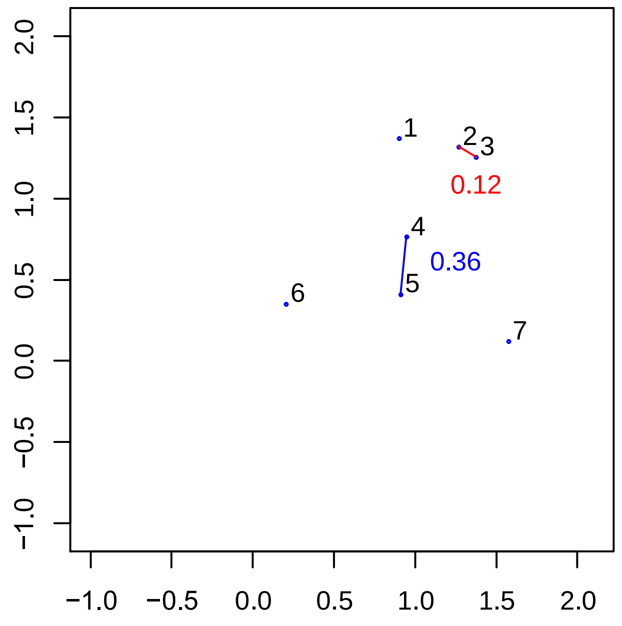
Hierarchical agglomerative clustering: Start with all data points in their own groups, and repeatedly merge groups, based on linkage function. Stop when points are in one group (this is agglomerative; there is also divisive)

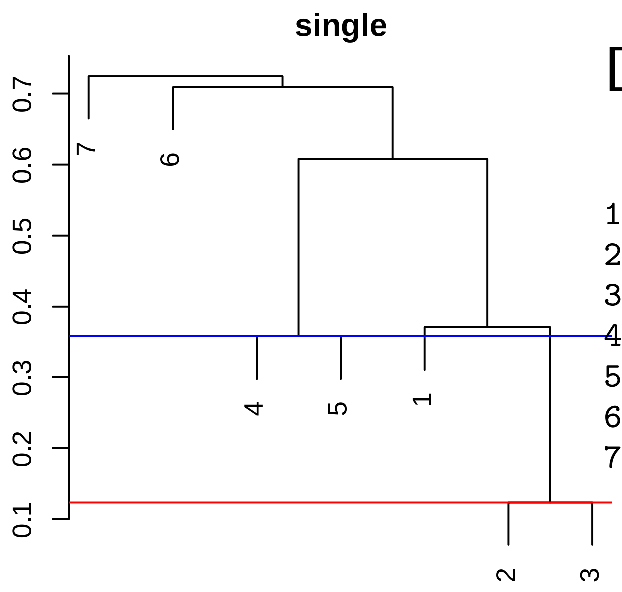
This produces a sequence of clustering assignments, visualized by a dendrogram (i.e., a tree). Each node in the tree represents a group, and its height is proportional to the linkage distance of its daughters

Three most common linkage functions: single, complete, average linkage. Single linkage measures the least dissimilar pair between groups, complete linkage measures the most dissimilar pair, average linkage measures the average dissimilarity over all pairs

Each linkage has its strengths and weaknesses

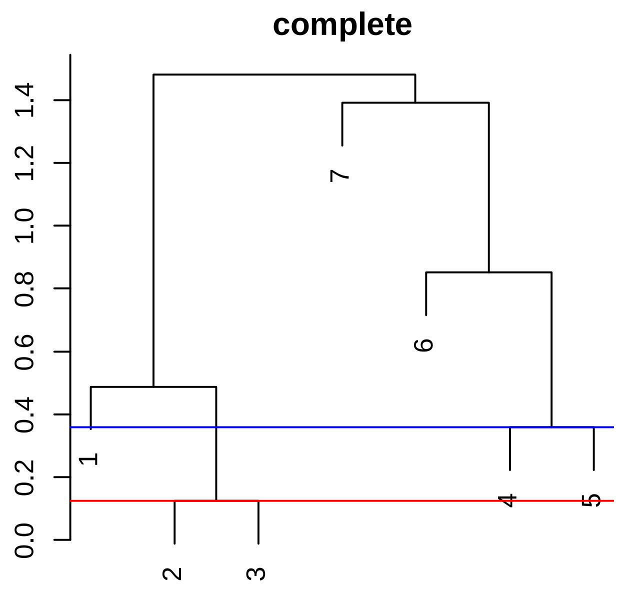
Careful Example

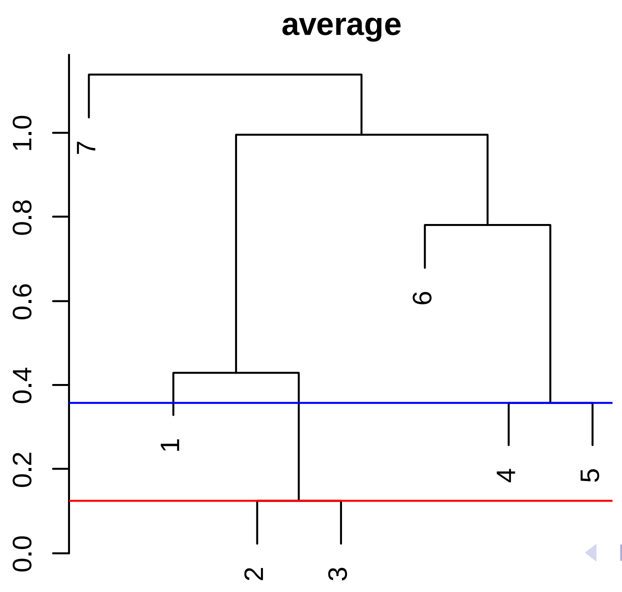




Distance matrix (Δ)

	1	2	3	4	5	6	7
1	0.00	0.37	0.49	0.61	0.96	1.24	1.42
2	0.37	0.00	0.12	0.64	0.98	1.44	1.24
3	0.49	0.12	0.00	0.65	0.97	1.48	1.15
4	0.61	0.64	0.65	0.00	0.36	0.85	0.90
5	0.96	0.98	0.97	0.36	0.00	0.71	0.72
6	1.24	1.44	1.48	0.85	0.71	0.00	1.39
7	1.42	1.24	1.15	0.90	0.72	1.39	0.00





(All Merging $\{1\}$ and $\{2,3\}$)

Single:	0.37
Complete:	0.49
Average:	$(0.37 + 0.49)/2 = 0.43$

(Next Agglomeration)

Single:	Merging $\{4,5\}$ & $\{1,2,3\}$ 0.61
Complete:	Merging $\{4,5\}$ & $\{6\}$ 0.85
Average:	Merging $\{4,5\}$ & $\{6\}$ $(0.85+0.71)/2 = 0.78$

Another linkage

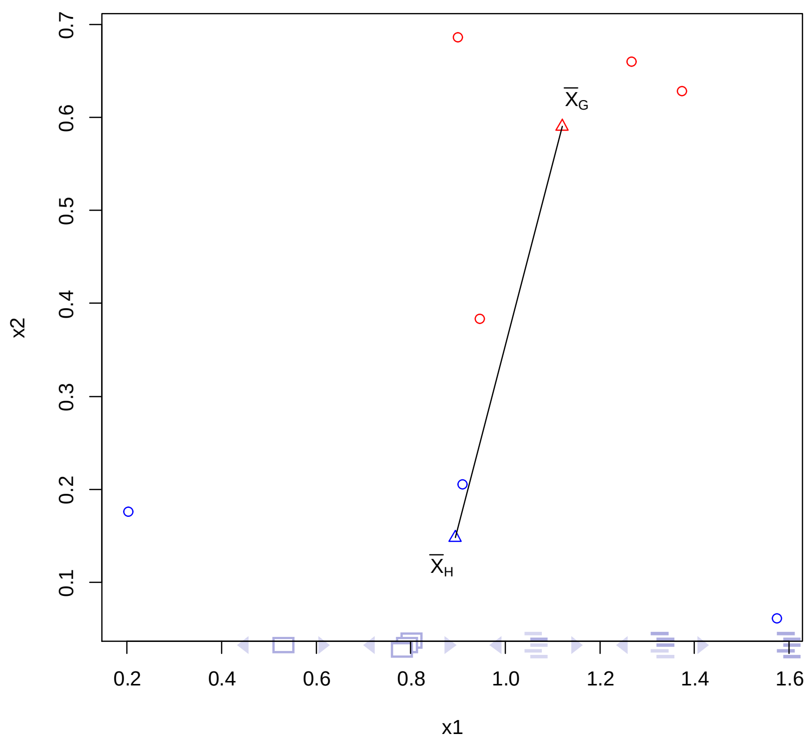
Centroid linkage is a commonly used and relatively new approach. Assume

- $X_i \in \mathbb{R}^p$
- $d_{ij} = ||X_i - X_j||_2^2$

Let $\overline{X}_G$ and $\overline{X}_H$ denote group averages for G, H. Then

$$d_{\text{centroid}} = ||\overline{X}_G - \overline{X}_H||_2^2$$

Example: There are two clusters (red and blue). The centroid linkage score ($d_{\text{centroid}}(G, H)$) is the distance between the centroids (black line segment).



Centroid linkage

Centroid linkage is

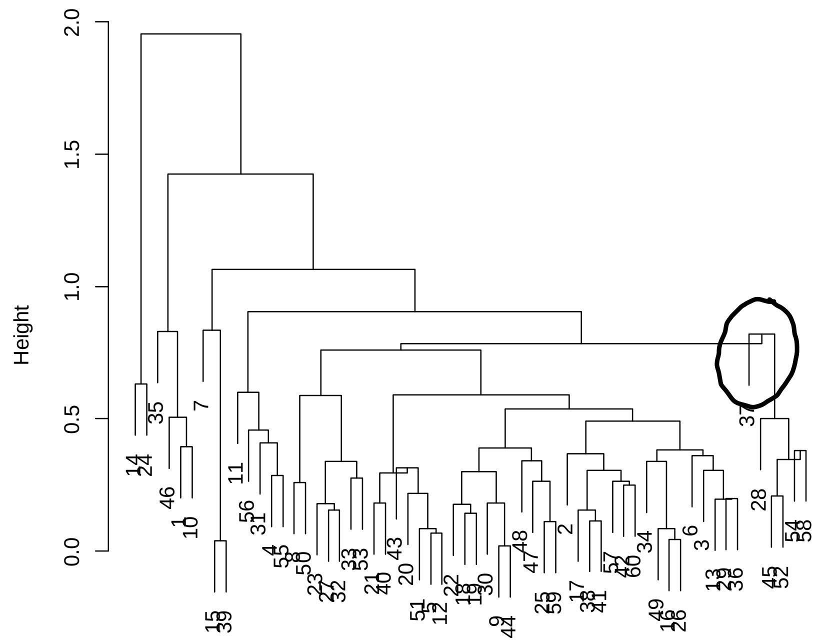
- ... quite intuitive
- ... widely used
- ... nicely analogous to K-means.
- ... very related to average linkage (and much, much faster)

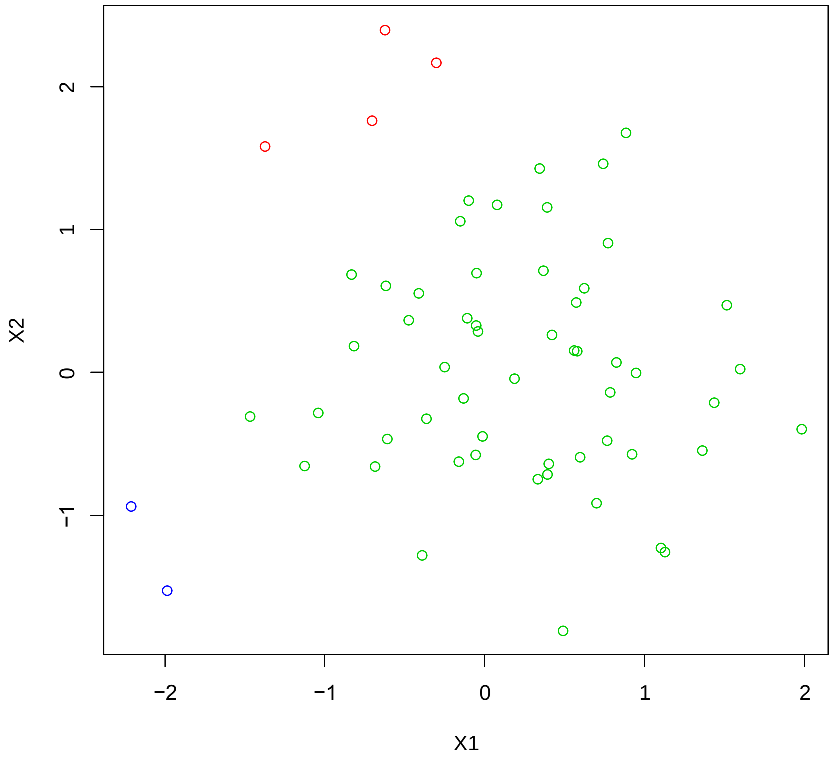
However, it has a very unsavory feature: inversions.

An inversion is when an agglomeration doesn't reduce the linkage distance.

Centroid linkage example

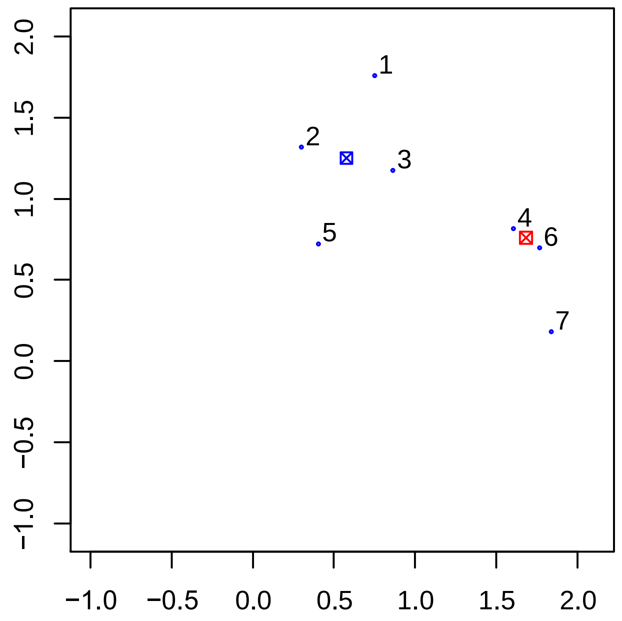
Same data as before. We can't look at cutting the tree, but we can still look at a 3 cluster solution.

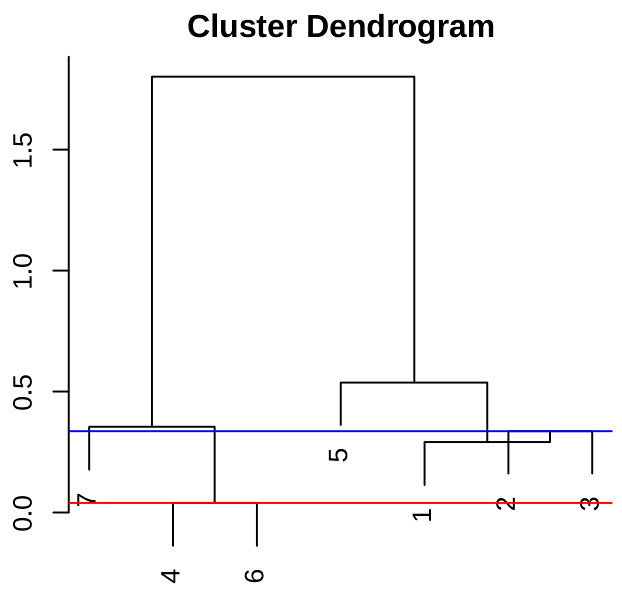




Cut interpretation: Even if there are no inversions, there still is no cut interpretation.

Careful Example: Steps 1,2,3





Distance matrix (Δ)

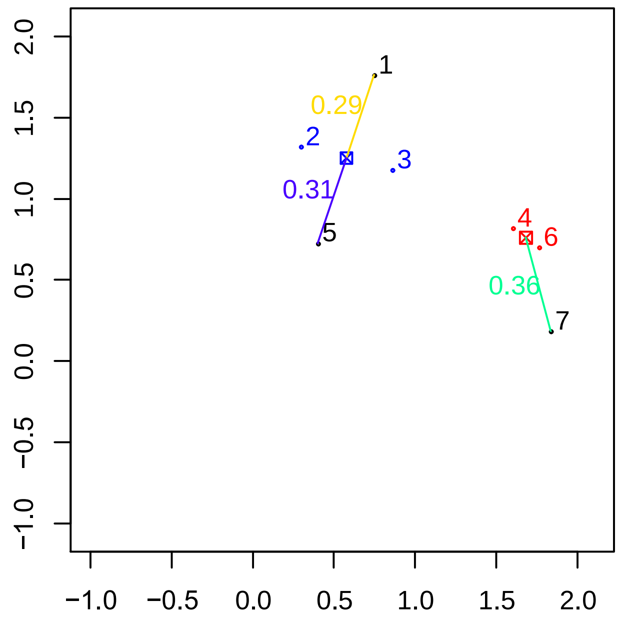
	1	2	3	4	5	6	7
1	0.00	0.40	0.35	1.62	1.20	2.16	3.67
2	0.40	0.00	0.34	1.96	0.37	2.54	3.66
3	0.35	0.34	0.00	0.68	0.42	1.05	1.94
4	1.62	1.96	0.68	0.00	1.45	0.04	0.46
5	1.20	0.37	0.42	1.45	0.00	1.86	2.35
6	2.16	2.54	1.05	0.04	1.86	0.00	0.27
7	3.67	3.66	1.94	0.46	2.35	0.27	0.00

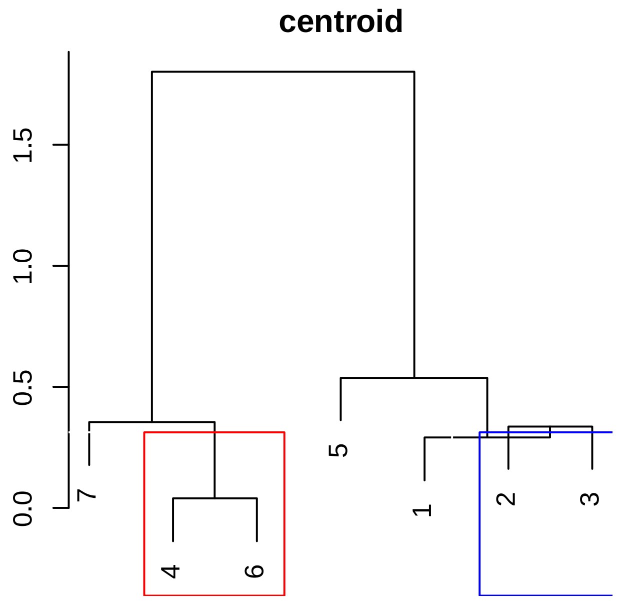
(This is squared Euclidean distance)

Centroid(4,6) = (1.68,0.76)

Centroid(2,3) = (0.58,1.25)

Careful Example: Step 4





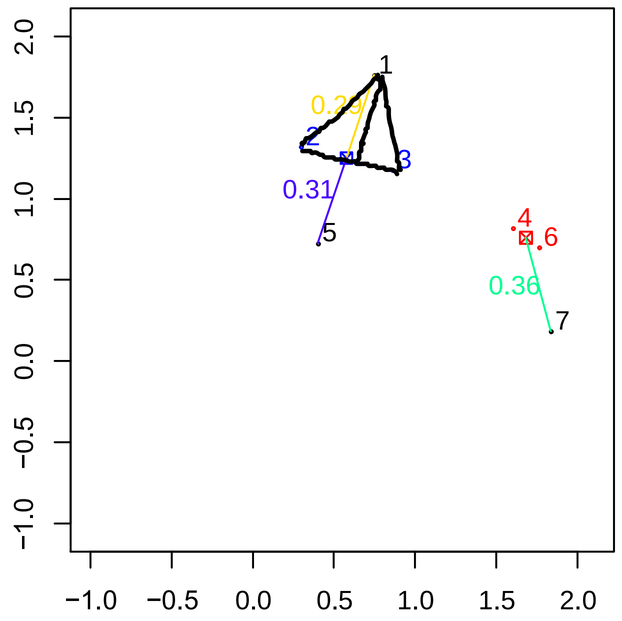
Distance matrix (Δ)

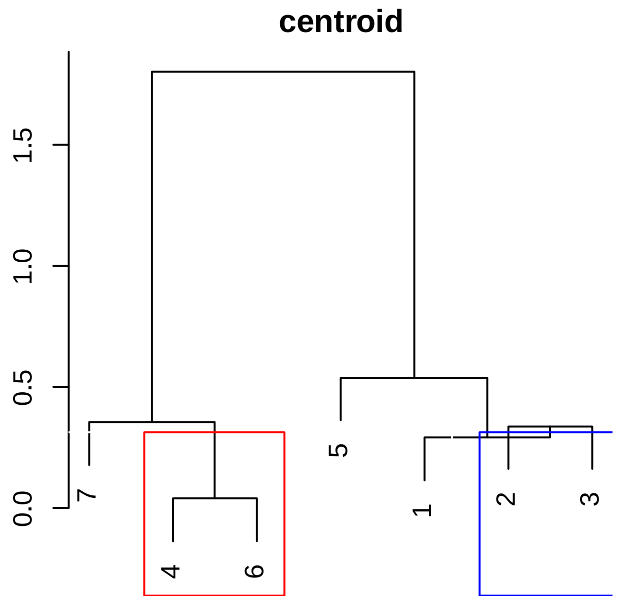
Which one gets merged?

```
     1    2    3    4    5    6    7
1 0.00 0.40 0.35 1.62 1.20 2.16 3.67
2 0.40 0.00 0.34 1.96 0.37 2.54 3.66
3 0.35 0.34 0.00 0.68 0.42 1.05 1.94
4 1.62 1.96 0.68 0.00 1.45 0.04 0.46
5 1.20 0.37 0.42 1.45 0.00 1.86 2.35
6 2.16 2.54 1.05 0.04 1.86 0.00 0.27
7 3.67 3.66 1.94 0.46 2.35 0.27 0.00
```

(This is squared Euclidean distance)

Careful Example: Step 4





Distance matrix (Δ)

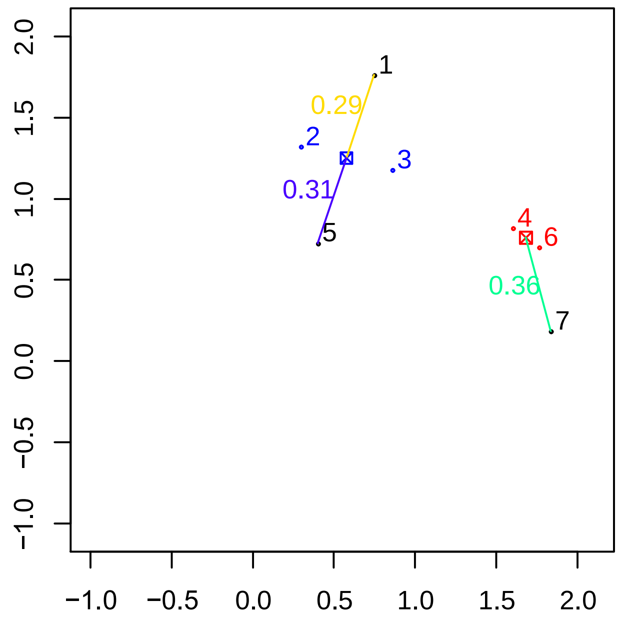
```
     1    2    3    4    5    6    7
1 0.00 0.40 0.35 1.62 1.20 2.16 3.67
2 0.40 0.00 0.34 1.96 0.37 2.54 3.66
3 0.35 0.34 0.00 0.68 0.42 1.05 1.94
4 1.62 1.96 0.68 0.00 1.45 0.04 0.46
5 1.20 0.37 0.42 1.45 0.00 1.86 2.35
6 2.16 2.54 1.05 0.04 1.86 0.00 0.27
7 3.67 3.66 1.94 0.46 2.35 0.27 0.00
```

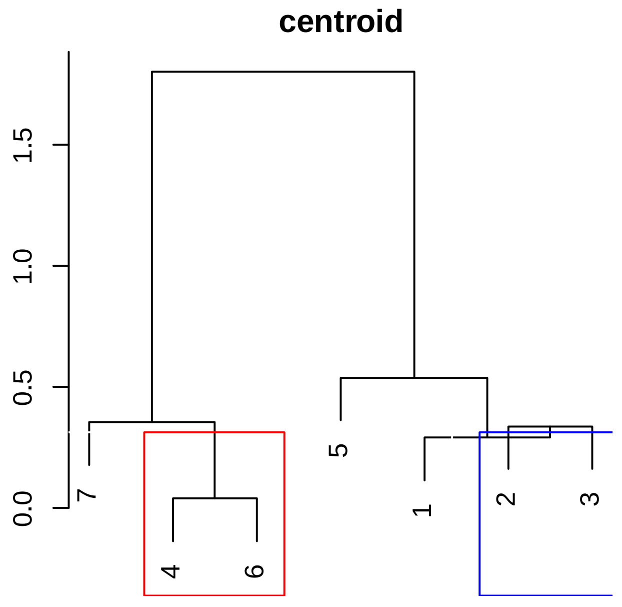
(This is squared Euclidean distance)

Which one gets merged?

($\{1\}$ and $\{2,3\}$)

Careful Example: Step 4





Distance matrix (Δ)

```
     1    2    3    4    5    6    7
1 0.00 0.40 0.35 1.62 1.20 2.16 3.67
2 0.40 0.00 0.34 1.96 0.37 2.54 3.66
3 0.35 0.34 0.00 0.68 0.42 1.05 1.94
4 1.62 1.96 0.68 0.00 1.45 0.04 0.46
5 1.20 0.37 0.42 1.45 0.00 1.86 2.35
6 2.16 2.54 1.05 0.04 1.86 0.00 0.27
7 3.67 3.66 1.94 0.46 2.35 0.27 0.00
```

(This is squared Euclidean distance)

Which one gets merged?

($\{1\}$ and $\{2, 3\}$)

```
method = 'centroid'
out = hclust(Delta,method=method)
rect.hclust(out,k=5,
            border=c('white','red','white','white','blue'))
```

Linkages summary

	No inversions?	Unchanged w/ monotone transformation?	Cut interpretation?	Notes
Single	✓	✓	✓	chaining
Complete	✓	✓	✓	crowding
Average	✓	X	X	
Centroid	X	X	X	inversions

Final notes:

- None of this helps determine what is the best linkage
- Use the linkage that seems the most appropriate for the types of clusters you want to get

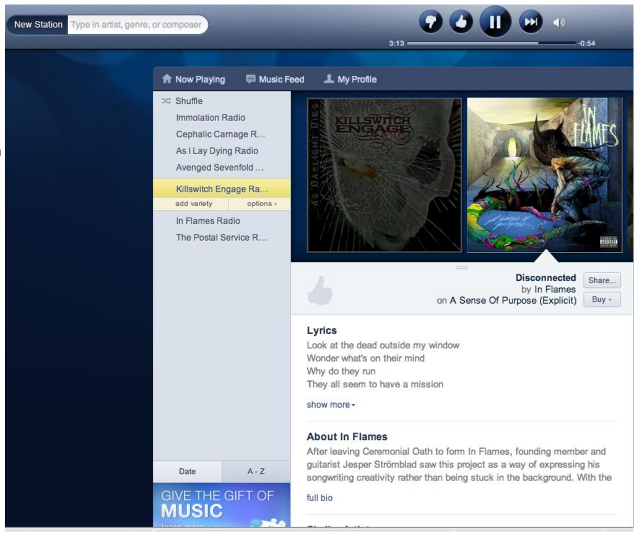
Designing a clever radio system

We have a lot of songs and dissimilarity scores between them (d_{ij})

We want to build a clever radio system that takes a song specified by the user and produces a song of the "same" type

We ask the user how "risky" he or she wants to be

1) CHOOSE LINKAGE
   ↳ COMPLETE
2) SPECIFY # CLUSTERS
   ↳ CUT TREE AT $h =$ "risky"
3) LISTEN TO SONG $i \in G$
4) CHOOSE A SONG RANDOMLY IN G. $d_{ij} \leq$ "risky" FOR ALL $j \in G$



Disconnected
by In Flames
on A Sense Of Purpose (Explicit)

Lyrics
Look at the dead outside my window
Wonder what's on their mind
Why do they run
They all seem to have a mission

About In Flames
After leaving Ceremonial Oath to form In Flames, founding member and guitarist Jesper Strömblad saw this project as a way of expressing his songwriting creativity rather than being stuck in the background. With the

1) SINGLE
2) CUT AT $h =$ "risky"
3) $i \in G$
4) $j \in G$

How can we use hierarchical clustering and with what linkage?

Linkages summary: Cut interpretations

Suppose we cut the tree at height $h = 1$.

Single	For each point X_i, there is another point X_j in the same cluster with $d_{ij} \leq 1$ (assuming more than 1 point in cluster). Also, no points in different clusters are closer than 1.
Complete	For each point X_i, every other point X_j in the same cluster has $d_{ij} \leq 1$.

Data analysis example

Diffuse large B-cell lymphoma (DLBCL) is the most common type of non-Hodgkin's lymphoma

It is clinically heterogeneous:

- 40% of patients respond well
- 60% of patients succumb to the disease

The researchers propose that this difference is due to unrecognized molecular heterogeneity in the tumors

We examine the extent to which genomic-scale gene expression profiling can further the understanding of B-cell malignancies.

Data analysis example

Here, we have gene expression data at 2,000 genes for 62 cancer cells.

There are 3 cancer diagnoses: FL, CLL, DLBCL. Each corresponds to a type of malignant lymphoma.

We want to use hierarchical clustering to understand this data set better.

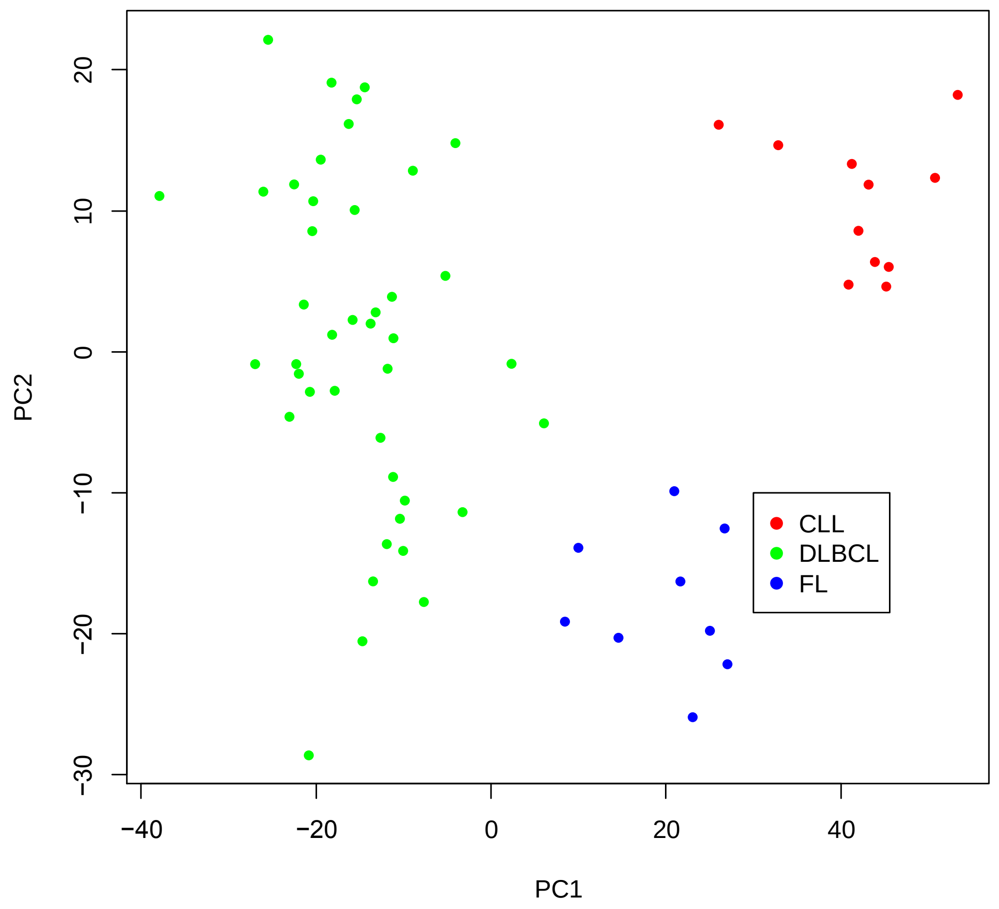
```
load('../data/alizadeh.RData')

genesT   = alizadeh$x
genes    = t(genesT)
Yfull    = alizadeh$type
Y        = as.vector(Yfull)
Y[Yfull == "DLBCL-A"] = 'DLBCL'
Y[Yfull == "DLBCL-G"] = 'DLBCL'
Y        = as.factor(Y)
dist.mat     = dist(genes)
```

PCA PLOT

$$X \in \mathbb{R}^{\overset{\text{OBS}}{n} \times \overset{\text{GENES}}{p}}$$

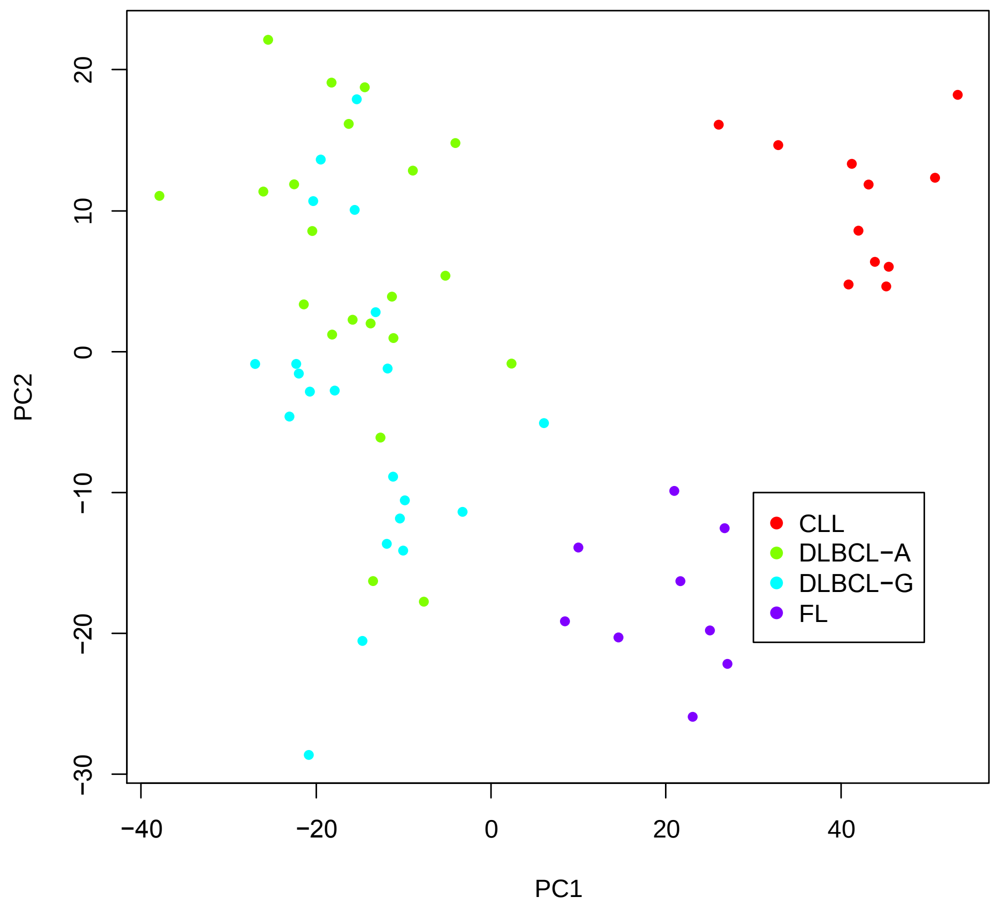
$$[\qquad\qquad\qquad]$$



Two clear groups for FL and CLL

DLBCL somewhat appears to be 1 group, but it is much more diffuse.

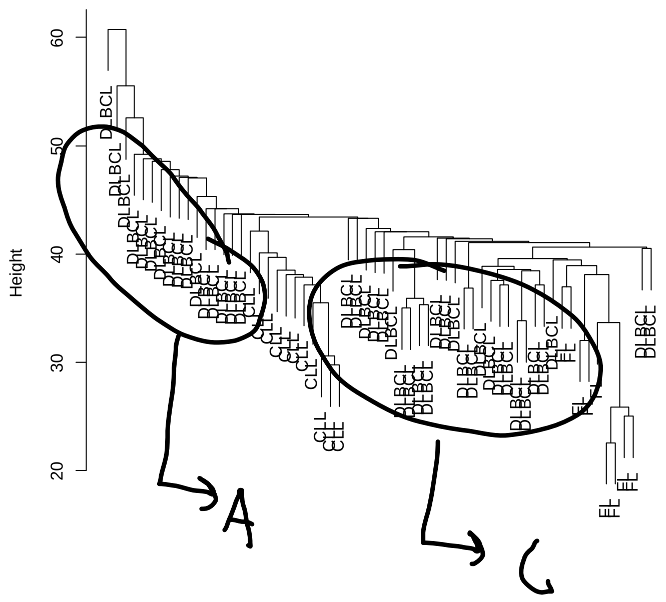
PCA PLOT



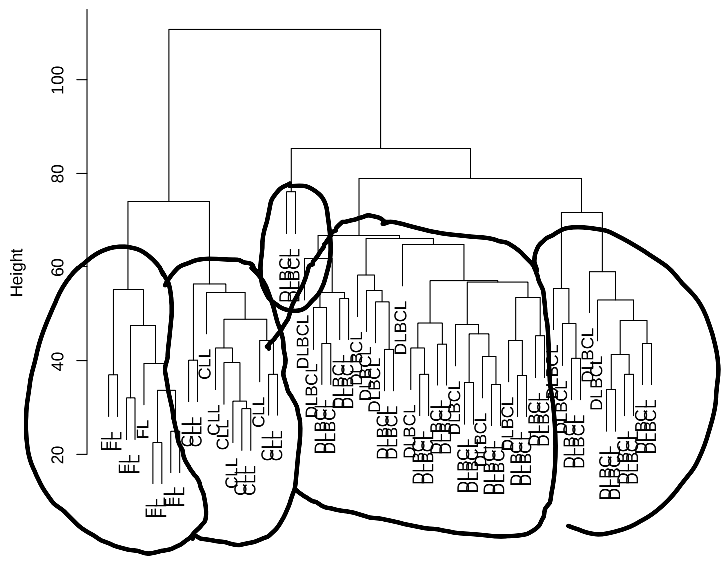
Here are the two sub-types identified by the researchers

Let's look at their results further.

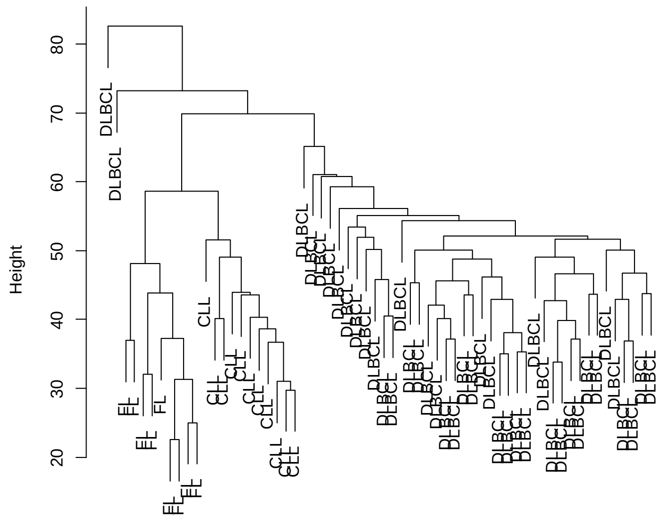
Four hierarchical cluster solutions



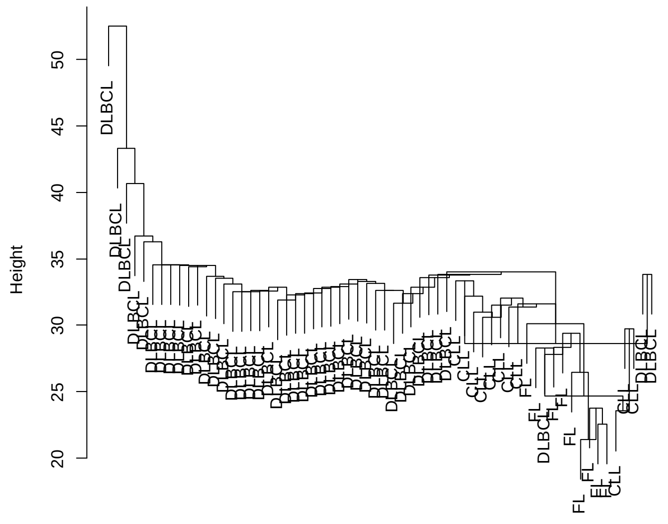
Single



Complete

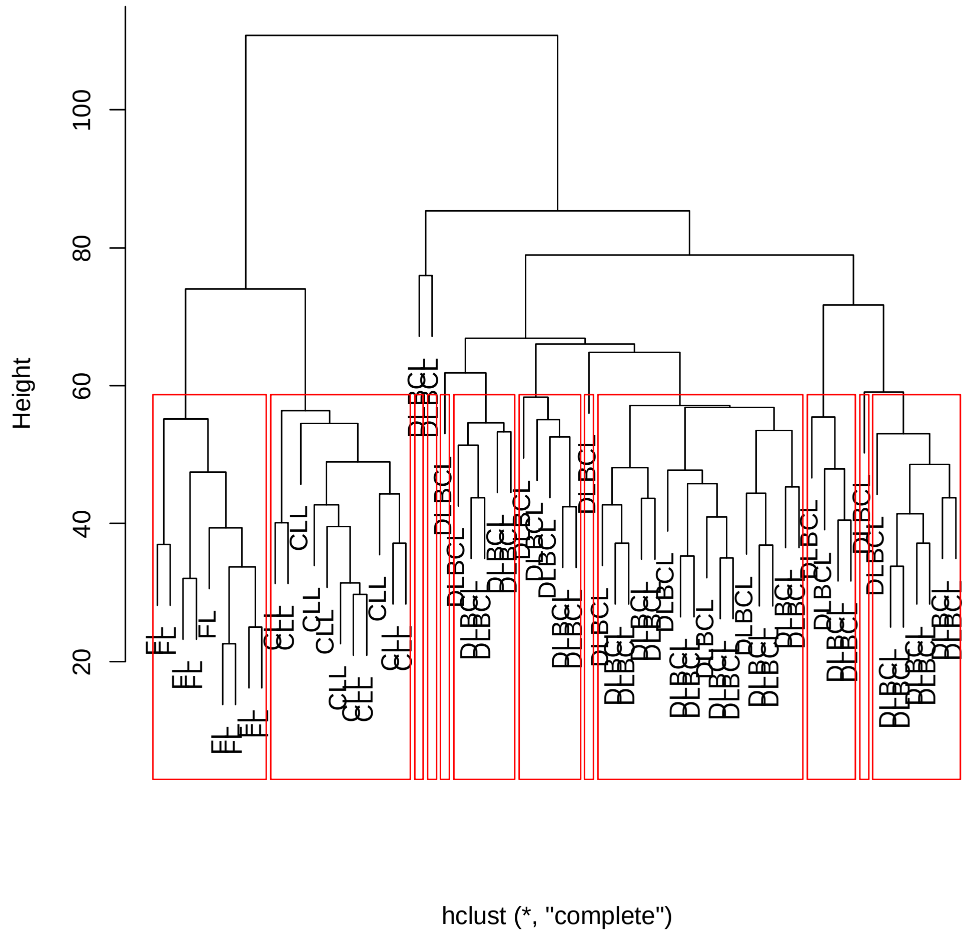


Average



Centroid

Complete linkage: A closer look



```
out.com = hclust(dist.mat,
      method='complete')
plot(out.com,xlab='',
    main='',labels=Y)
rect.hclust(out.com,k=12)

out.cut = cutree(out.com,
           k=12)
```

Notice that FL and CLL are distinctly grouped, while there are many clusters inside the DLBCL type.