Big Data Analytics

Session 9

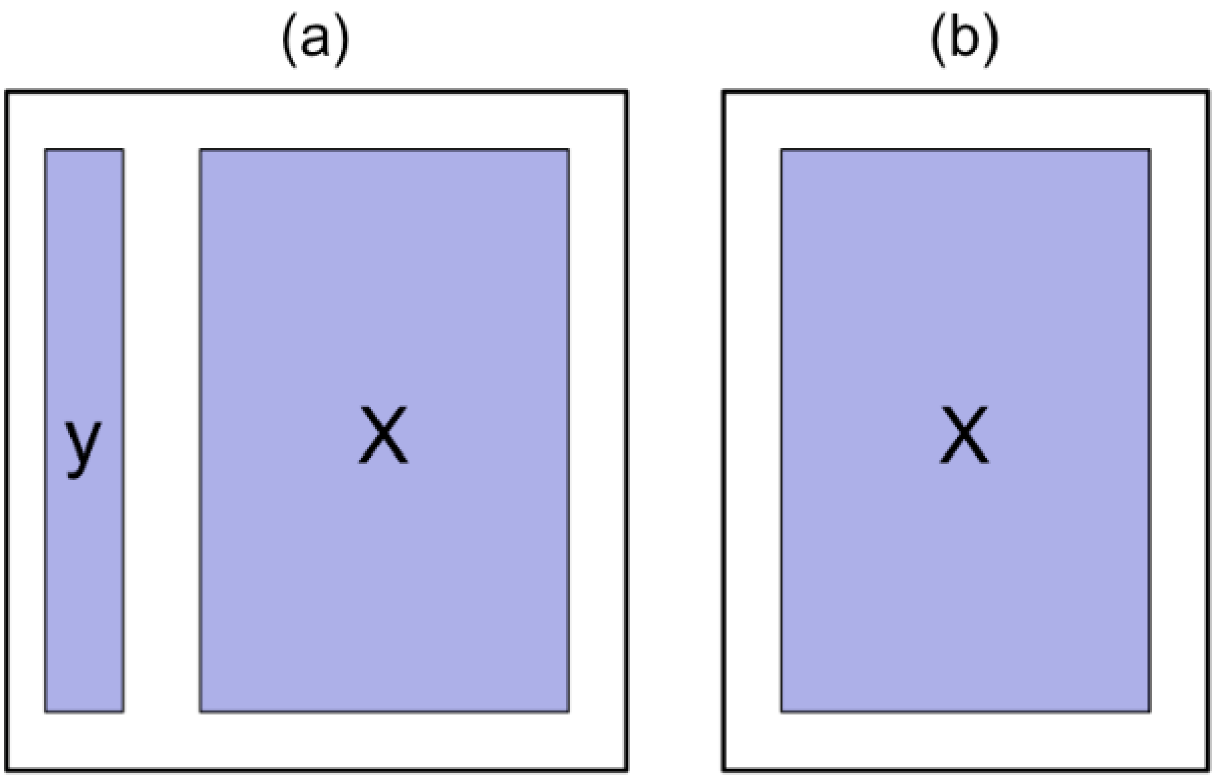
Clustering

Supervised Learning

- Tools:
  - Regression: linear regression,
    regression trees (+ bagging, random forests)
  - Classification: logistic regression,
    classification trees (+ bagging, random forests),
    SVM
- Assessments:
  - The quality of the results obtained:
    cross validation,
    validation on an independent test set, etc

Supervised vs. Unsupervised Learning

- Supervised Learning: both X and y are known
- Unsupervised Learning: only X



Supervised Learning Unsupervised Learning

Unsupervised Learning Examples

- Widely used in a number of fields
  - Cancer research
    - Goal: Assay gene expression levels in 100 patients with breast cancer
    - Solution: Look for subgroups among the breast cancer sample, or among the genes, in order to obtain a better understand of the disease
  - Online recommender system
    - Goal: To show the items in which a customer is particularly likely to be interested
    - Solution: Identify groups of shoppers with similar browsing and purchase histories, as well as items that are of particular interest to the shoppers within each group
  - Search engine
    - Goal: Choose what to display to an individual
    - Solution: Choose based on the click histories of other individuals with similar search patterns

Unsupervised Learning

- More challenging:
  - more subjective
  - no simple goal for the analysis: e.g, prediction of a response
  - hard to assess:
    - no way to check, as the true answer is unknown → unsupervised!
- Approaches to unsupervised learning
  - Clustering
    - K-means, hierarchical clustering (today)
  - Hidden Markov Models
  - Feature extraction for dimension reduction
    - Principal component analysis (next week)

Outline

- What is Clustering?
- K-Means Clustering
- Hierarchical Clustering
- Final Thoughts

Clustering

- Clustering refers to a set of techniques for finding subgroups, or clusters, in a dataset

- A good clustering is one when
  - the observations within a group are similar
  - but between groups are very different

- For example, suppose we collect p measurements on each of n breast cancer patients. There may be different unknown types of cancer which we could discover by clustering the data

Different Clustering Methods

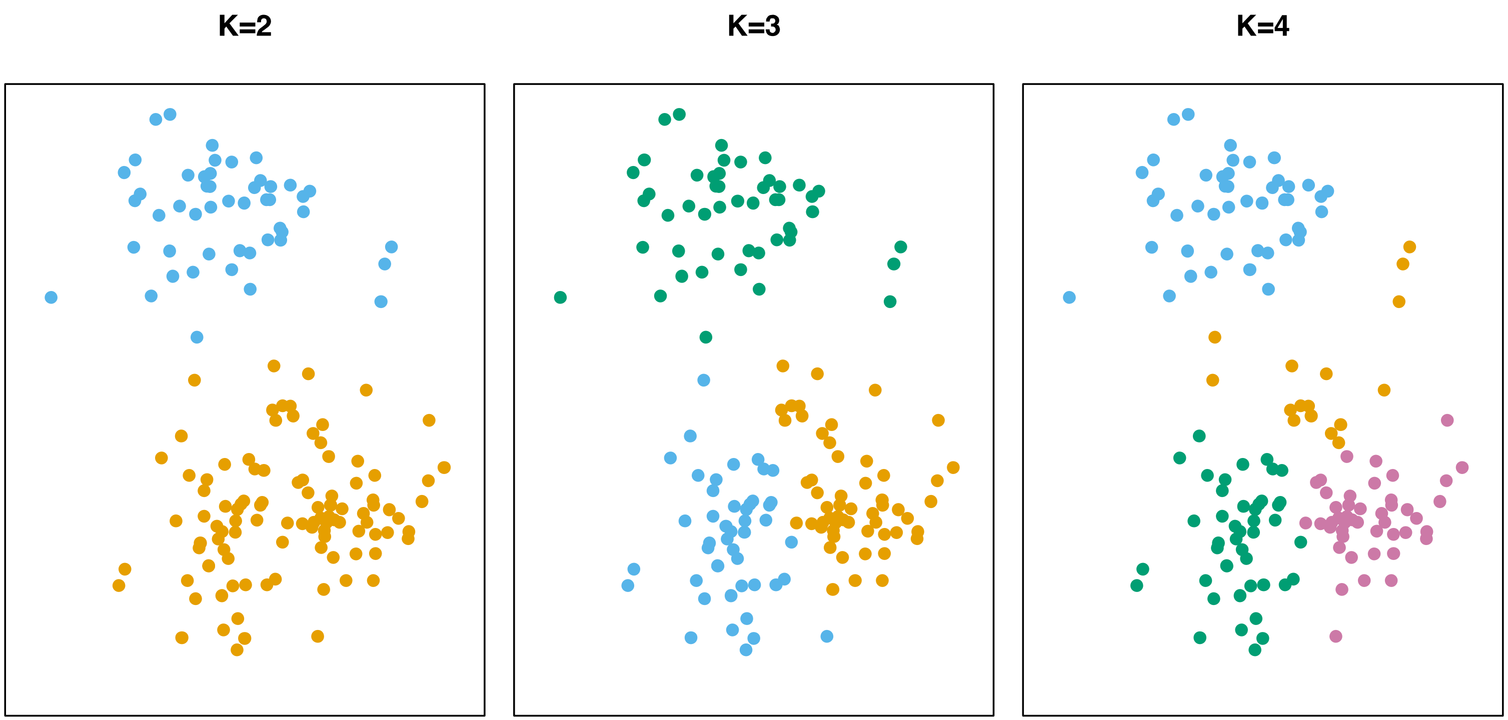
- There are many different types of clustering methods
- We will concentrate on two of the most commonly used approaches
  - K-Means Clustering
  - Hierarchical Clustering

Outline

- What is Clustering?
- K-Means Clustering
- Hierarchical Clustering
- Final Thoughts

K-Means Clustering

- To perform K-means clustering, one must first specify the desired number of clusters K
- Then the K-means algorithm will assign each observation to exactly one of the K clusters



How does K-Means work?

- We would like to partition that dataset into K clusters

$$C_1, \ldots, C_K$$

  - Each observation belongs to at least one of the K clusters
  - The clusters are non-overlapping, i.e. no observation belongs to more than one cluster → Each observation belongs to at most one of the K clusters

  ➔ Each observation belongs to exactly one of the K clusters

- The objective is to have a minimal "within-cluster-variation", i.e. the elements within a cluster should be as similar as possible

- One way of achieving this is to minimise the sum of all the pairwise squared Euclidean distances between the observations in each cluster

Euclidean Distance

Euclid of Alexandria

- The Euclidean distance between point **s** and **t** is the length of the line segment connecting them
  - One dimensional: d(s, t) = $\sqrt{(s-t)^2}$ = | s – t |
    - s = 0.2, t = -2.8 ➔ d(s, t) = 3
  - Two dimensional: d(**s**, **t**) = d((s_1, s_2), (t_1, t_2)) = $\sqrt{(s_1-t_1)^2+(s_2-t_2)^2}$
    - **s** = (1, 6.5), **t** = (4, 2.5) ➔ d(**s**, **t**) = $\sqrt{(1-4)^2+(6.5-2.5)^2}$ = 5

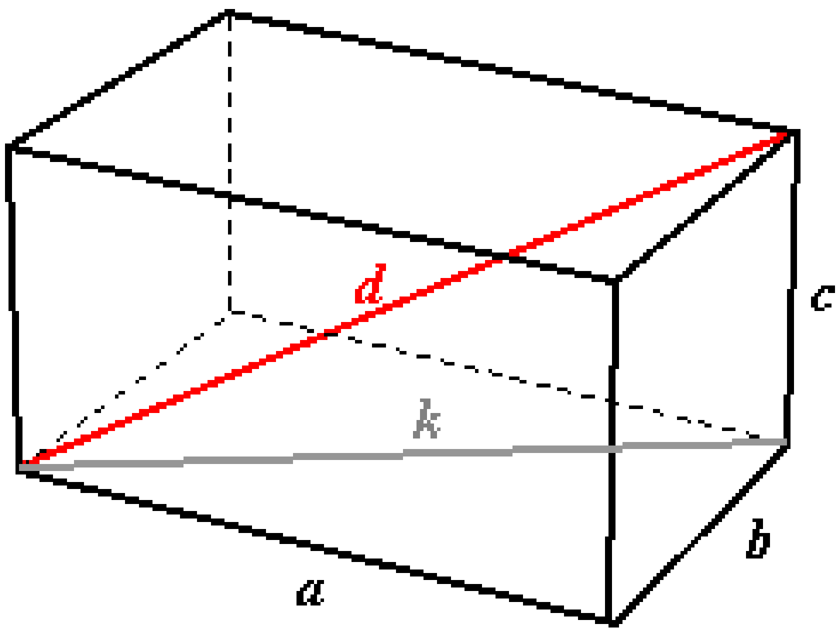
Pythagorean theorem

  - Three dimensional: d(**s**, **t**) = $\sqrt{(s_1-t_1)^2+(s_2-t_2)^2+(s_3-t_3)^2}$
    - **s** = (0.2, -1.5, 0), **t** = (-2.8, 2.5, 75)

    ➔d(**s**, **t**) = $\sqrt{(0.2-(-2.8))^2+(-1.5-2.5)^2+(0-75)^2}$ = 10

3-Dimensional Euclidean Distance

$$d^2 = a^2 + b^2 + c^2$$



Because:

$$k^2 = a^2 + b^2$$

$$d^2 = k^2 + c^2 = a^2 + b^2 + c^2$$

Euclidean Distance


Euclid of Alexandria

- The Euclidean distance between point **s** and **t** is the length of the line segment connecting them
  - One dimensional: $d(s, t) = \sqrt{(s-t)^2} = |s - t|$
    - $s = 0.2$, $t = -2.8$ → $d(s, t) = 3$
  - Two dimensional: $d(\mathbf{s}, \mathbf{t}) = d((s_1, s_2), (t_1, t_2)) = \sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2}$
    - $\mathbf{s} = (1, 6.5)$, $\mathbf{t} = (4, 2.5)$ → $d(\mathbf{s}, \mathbf{t}) = \sqrt{(1-4)^2 + (6.5-2.5)^2} = 5$

Pythagorean theorem

  - Three dimensional: $d(\mathbf{s}, \mathbf{t}) = \sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2 + (s_3 - t_3)^2}$
    - $\mathbf{s} = (0.2, -1.5, 0)$, $\mathbf{t} = (-2.8, 2.5, 75)$

    → $d(\mathbf{s}, \mathbf{t}) = \sqrt{(0.2-(-2.8))^2 + (-1.5-2.5)^2 + (0-75)^2} = 10$

What is d(s,s)?

  - n dimensional: $d(\mathbf{s}, \mathbf{t}) = \sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2 + \ldots + (s_n - t_n)^2}$

- Squared Euclidean distance: $d^2(\mathbf{s}, \mathbf{t}) = (s_1 - t_1)^2 + (s_2 - t_2)^2 + \ldots + (s_n - t_n)^2$

Within-Cluster Variation

- Suppose we have 4 points (observations) in a cluster C, o_1, o_2, o_3, o_4, the sum of the pairwise squared Euclidean distances is

$$\text{sum}(C) = d^2(o_1, o_2) + d^2(o_1, o_3) + d^2(o_1, o_4) + d^2(o_2, o_3) + d^2(o_2, o_4) + d^2(o_3, o_4)$$

- The Within-Cluster Variation of cluster C is $W(C) = \dfrac{\text{sum}(C)}{\#C}$
  - #C is the number of observations in the cluster C
  - #C = 4 in this example

- Suppose we have K clusters, C_1, C_2, …, C_K, the task is to minimise the sum of all the within-cluster variations:

$$W(C_1) + W(C_2) + \ldots + W(C_K)$$

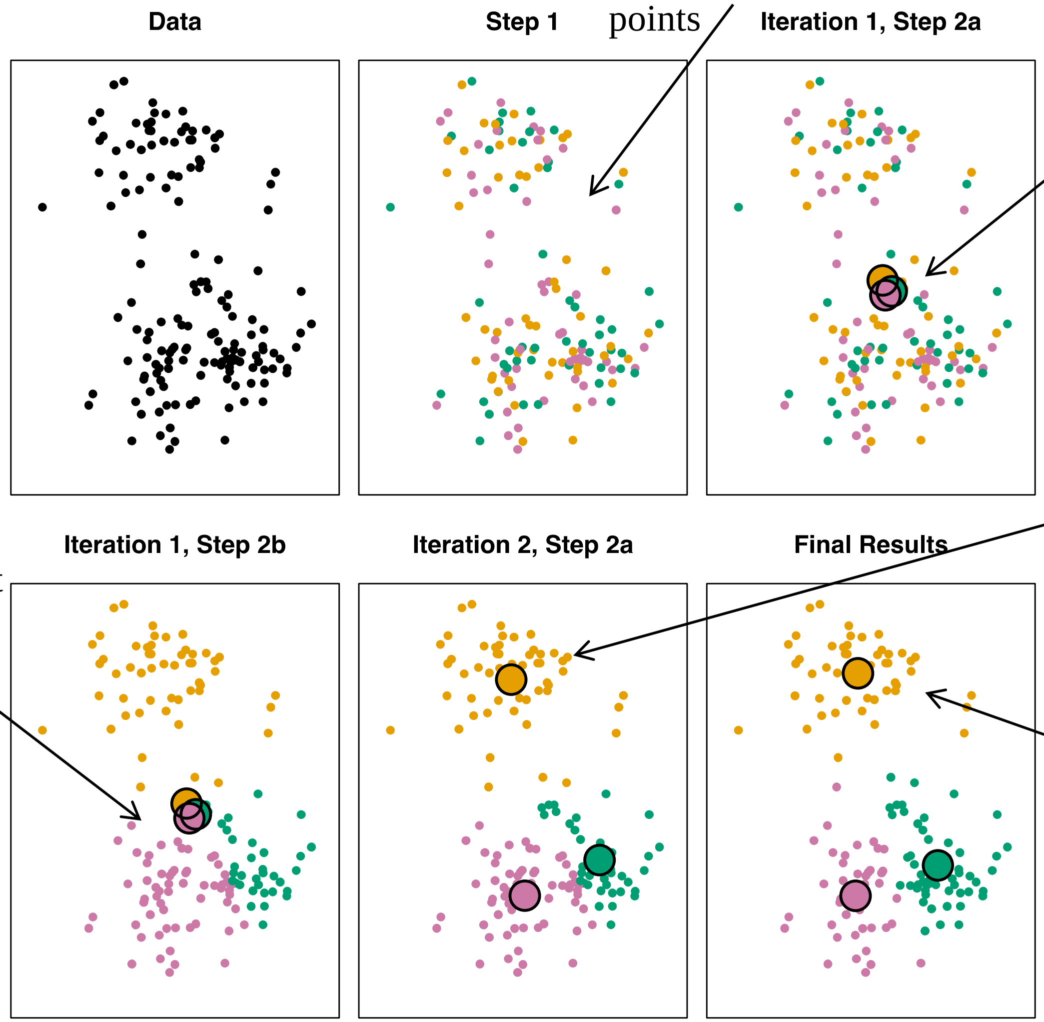
K-Means Algorithm

- Initial Step:
  - Randomly assign each observation to one of K clusters

- Iterate until the cluster assignments stop changing:
  - For each of the K clusters, compute the cluster centroid.
    - A cluster centroid is the mean of the observations assigned to the cluster
    - For example: In a cluster there are 2 observations with 4 features
      - First observation: (2, 4, 6, 8), Second observation: (2.2, 4.4, 6.6, 8.8)
      - Cluster centroid: (2.1, 4.2, 6.3, 8.4)
  - Assign each observation to the cluster whose centroid is closest
    - "closest" is defined using Euclidean distance

An Illustration of the K-Means Algorithm

K = 3



Random Assignment of points

Compute cluster centers from initial assignments

Computer new cluster centers

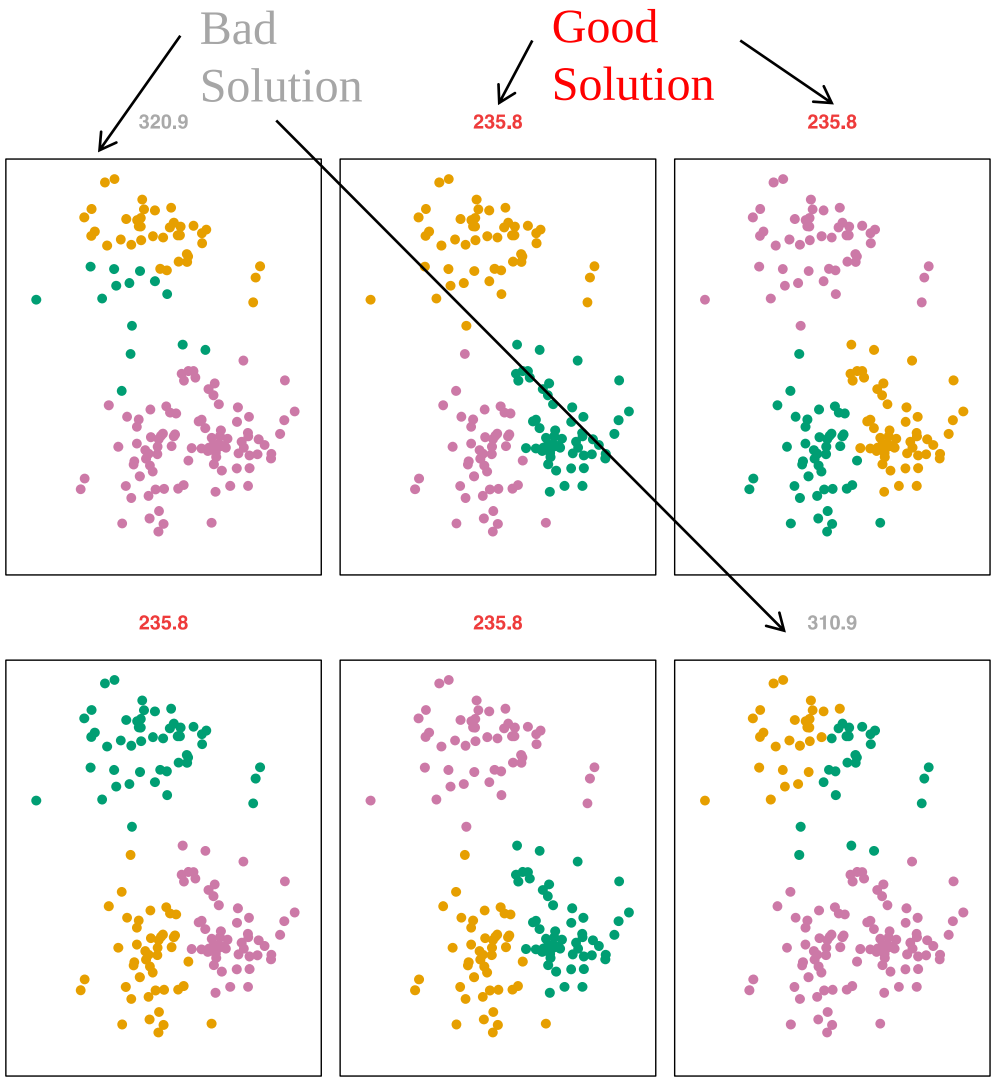
Assign points to closest cluster center

Now there is no further change so stop

Local Optimums

- Because the K-means algorithm finds a local rather global optimum, it can
  - get stuck in "local optimums" and
  - not find the best solution
  - The results obtained will depend on the initial (random) assignment of each observation.

- Hence, it is important to
  - run the algorithm multiple times
  - with random starting points
  - to find a good solution

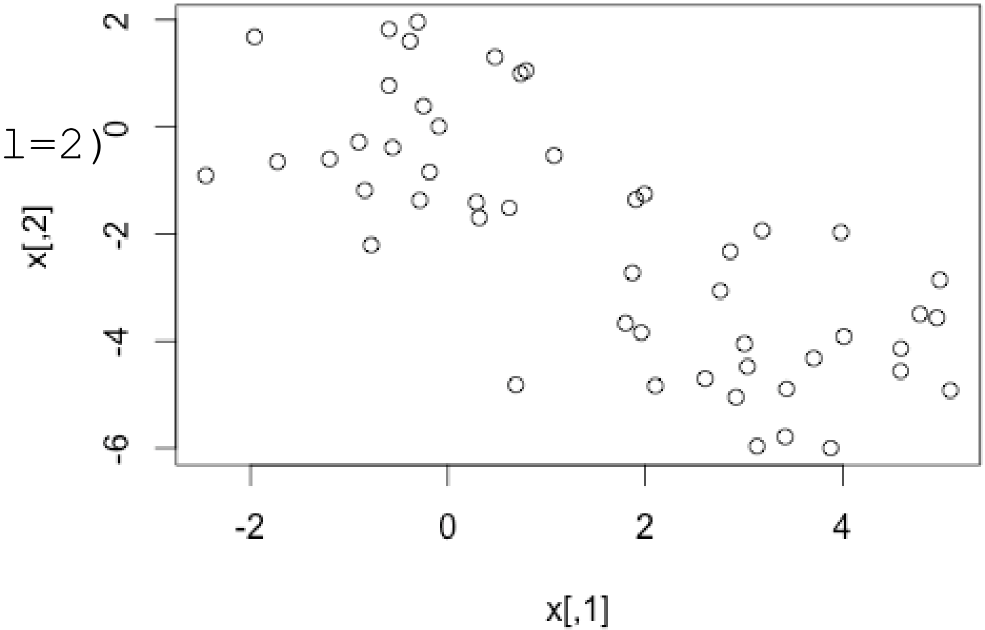


Numbers above each plot: sum of all the within-cluster variations

K-Means in R

- A simple simulated example in which there truly are two clusters in the data:
  - the first 25 observations have a mean shift relative to the next 25 observations:

```
set.seed(2)
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
plot(x)
```



K-Means in R

- The function `kmeans()` performs K-means clustering in R

```
km.out <- kmeans(x,2,nstart=20)
km.out$cluster
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
[26] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
plot(x,col=(km.out$cluster+1),
        main="K-Means Clustering Results with K=2",
        xlab="",ylab="",
        pch=20,cex=2)
```

cex number indicating the amount by which plotting text and symbols should be scaled relative to the default.

1=default,

1.5 is 50% larger,

0.5 is 50% smaller, etc

> palette()

> [1] "black" "red" "green3" "blue" "cyan" "magenta" "yellow" "gray"

K-Means in R

- What if we have guessed K = 3 instead?
  - For real data, in general we do not know the true number of clusters.

```
> set.seed(4)
> km.out <- kmeans(x,3,nstart=20)
> km.out
```

K-means clustering with 3 clusters of sizes 10, 23, 17

Cluster means:

	[,1]	[,2]
1	2.3001545	-2.69622023
2	-0.3820397	-0.08740753
3	3.7789567	-4.56200798

🡐 The centroids for three clusters

Clustering vector:

[1] 3 1 3 1 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3 1 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2

Within cluster sum of squares by cluster:

[1] 19.56137 52.67700 25.74089

🡐 withinss: $W(C_1)$ $W(C_2)$ $W(C_3)$

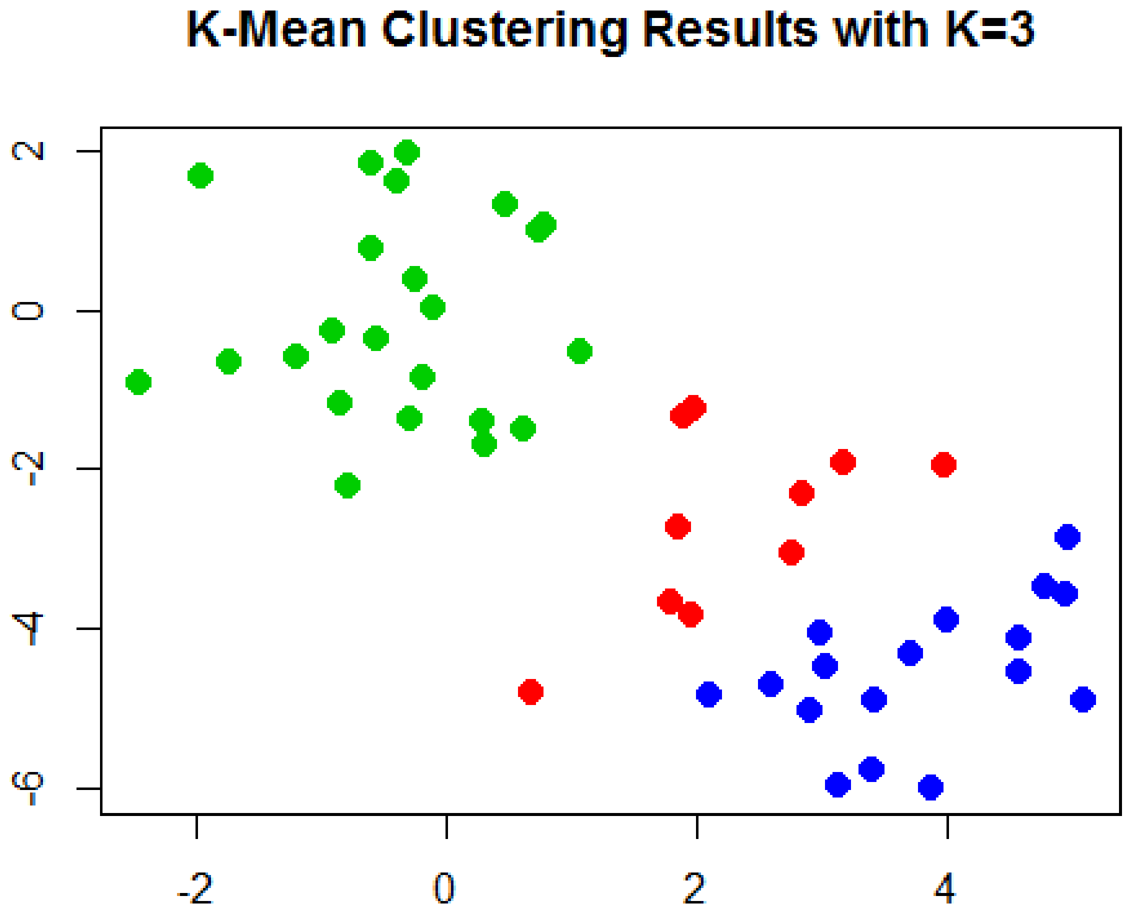
(between_SS / total_SS = 79.3 %)

Available components:

[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter" "ifault"

K-Means in R

```
plot(x,col=(km.out$cluster+1), xlab="", ylab="",
     main="K-Mean Clustering Results with K=3", pch=20, cex=2)
```



K-Means in R

- **Some explanations on the available components:**
  - "cluster": a vector of integers (which cluster a point belongs to)
    - km.out$cluster: 3 1 3 1 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3 1 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
  - "centers": The centroids of the clusters
  - "totss": The total sum of squares (km.out$totss = 473.6179)
  - "withinss": Individual within-cluster sum of squares
    - $W(C_1)$, $W(C_2)$, …, $W(C_K)$ (19.56137 52.67700 25.74089)
  - "tot.withinss": The total within-cluster sum of squares
    - $W(C_1) + W(C_2) + \ldots + W(C_K)$ (In R: sum(withinss)) (97.97927)
    - We aim to minimise this – the objective
  - "betweenss": The total between-cluster sum of squares (375.6386)
    - totss = tot.withinss + betweenss (473.6179 = 97.97927 + 375.6386)
  - "size": The number of points in each cluster (10 23 17)

K-Means in R

- To run the kmeans() function in R with multiple initial cluster assignments, we use the nstart argument.
  - n: How many times the initial cluster assignments will be set
  - kmeans() will report the best results

```
set.seed(3)
km.out <- kmeans(x,3,nstart=1)
km.out$tot.withinss
[1] 104.3319
km.out <- kmeans(x,3,nstart=20)
km.out$tot.withinss
[1] 97.97927
```
← Recall: we are to minimise tot.withinss

- Strongly recommend to use a large value of nstart to avoid local optimum.
- Important to set a random seed before performing K-means clustering
  - To replicate the initial cluster assignments as well as the K-means output.

Outline

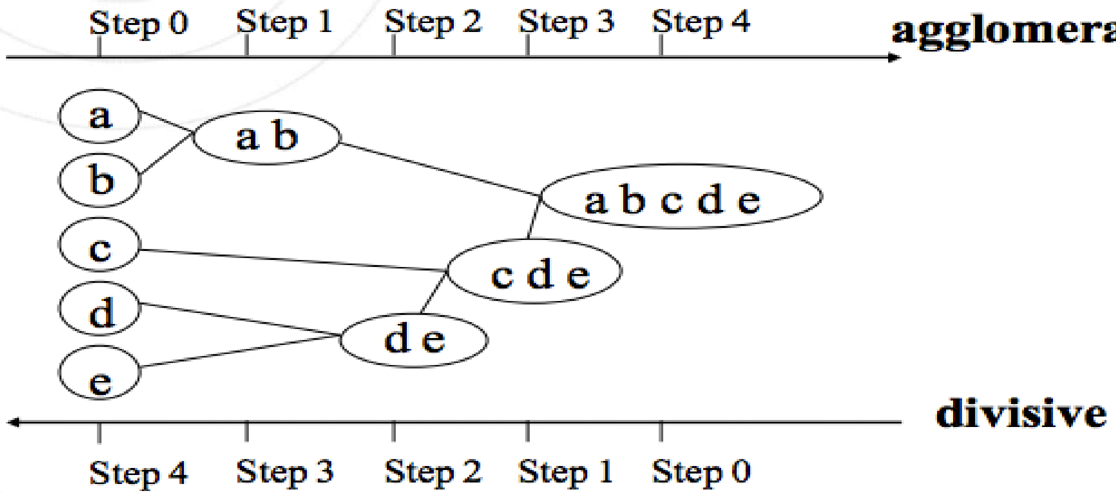
- What is Clustering?
- K-Means Clustering
- Hierarchical Clustering
- Final Thoughts

Hierarchical Clustering

- K-Means clustering requires choosing the number of clusters K.
- If we don't want to do that, an alternative is to use Hierarchical Clustering

Agglomerative vs Divisive

- Two types of hierarchical clustering algorithms
  - Agglomerative (i.e., bottom-up):
    - Start with all points in their own group
    - Until there is only one cluster, repeatedly: merge the two groups that have the smallest dissimilarity
  - Divisive (i.e., top-down):
    - Start with all points in one cluster
    - Until all points are in their own cluster, repeatedly: split the group into two resulting in the biggest dissimilarity



- Agglomerative strategies are simpler, we'll focus on them.
- Divisive methods are still important, but we won't be able to cover them in the lecture.

Hierarchical Agglomerative Clustering (HAC) Algorithm

- The dendogram is produced as follows:
  - Start with each point as a separate cluster (*n* clusters)
  - Calculate a measure of dissimilarity between all points/clusters
  - Fuse two clusters that are most similar so that there are now *n*-1 clusters
  - Fuse next two most similar clusters so there are now *n*-2 clusters
  - Continue until there is only 1 cluster

An Example of HAC Algorithm

Given these 9 observations, an agglomerative algorithm might decide on a clustering sequence as follows:

Start with 9 clusters

Step 1: {1}{2}{3}{4}{5}{6}{7}{8}{9}

Step 2: {1}{2}{3}{4}{5,7}{6}{8}{9}

Step 3: {1,6} {2}{3}{4}{5,7}{8}{9}

Step 4: {1,6} {2}{3}{4}{5,7,8}{9}

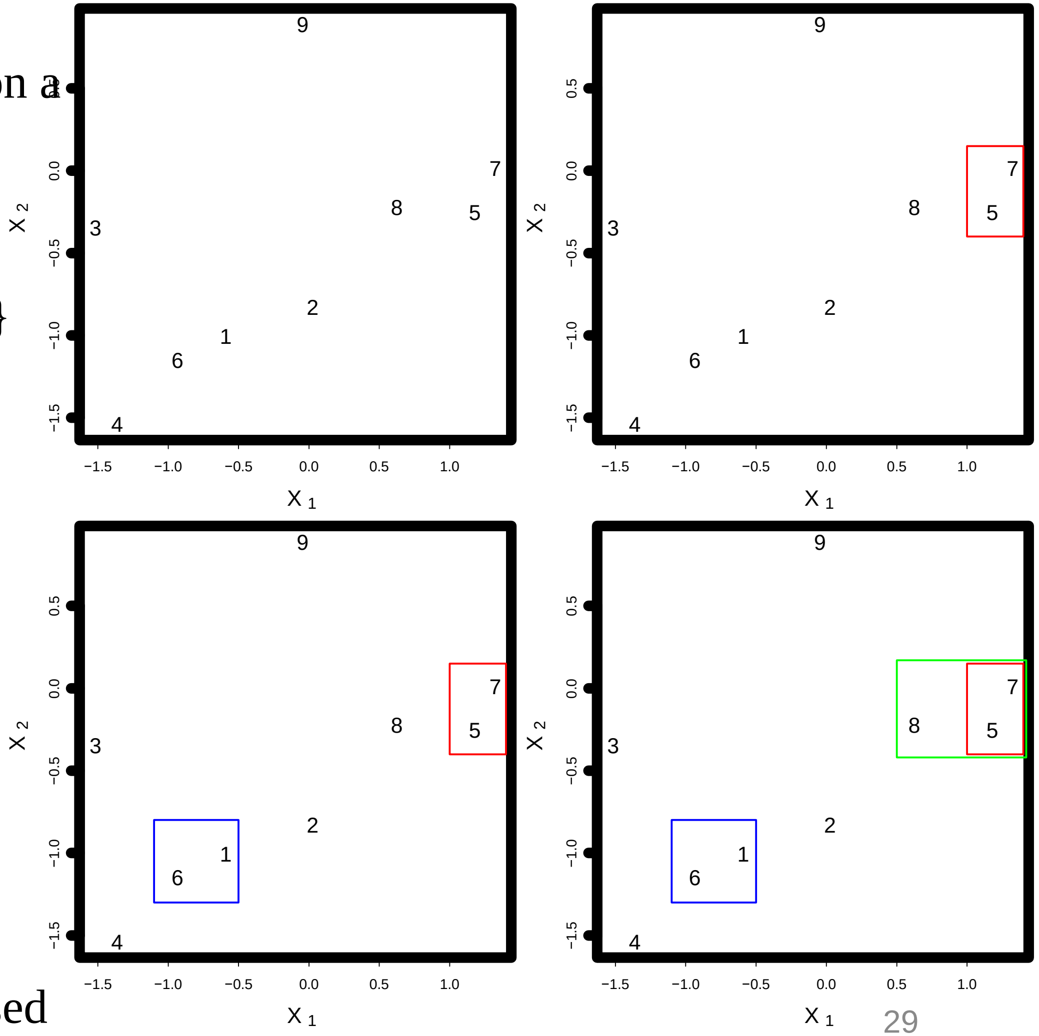
Step 5: {1,4,6} {2}{3}{5,7,8}{9}

Step 6: {1,3,4,6} {2}{5,7,8}{9}

Step 7: {1,3,4,6} {2,5,7,8}{9}

Step 8: {1,3,4,6} {2,5,7,8,9}

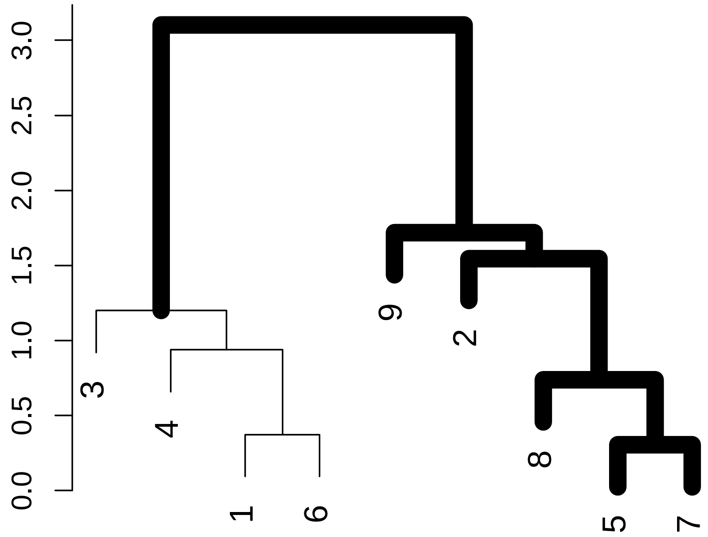
Step 9: {1,2,3,4,5,6,7,8,9}

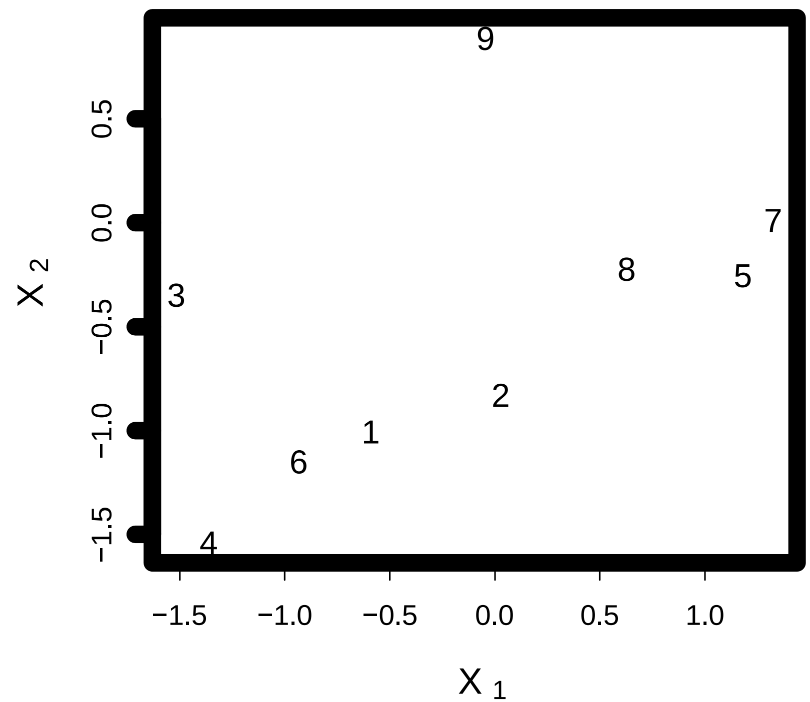


Terminate when all observations are fused

Dendrogram

- We can represent the sequence of clustering assignments as a dendrogram:





Step 1: {1}{2}{3}{4}{5}{6}{7}{8}{9}

Step 2: {1}{2}{3}{4}{5,7}{6}{8}{9}

Step 3: {1,6} {2}{3}{4}{5,7}{8}{9}

Step 4: {1,6} {2}{3}{4}{5,7,8}{9}

Step 5: {1,4,6} {2}{3}{5,7,8}{9}

Step 6: {1,3,4,6} {2}{5,7,8}{9}

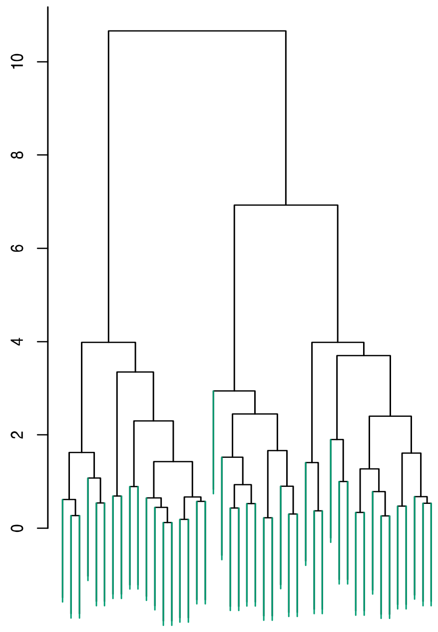
Step 7: {1,3,4,6} {2,5,7,8}{9}

Step 8: {1,3,4,6} {2,5,7,8,9}

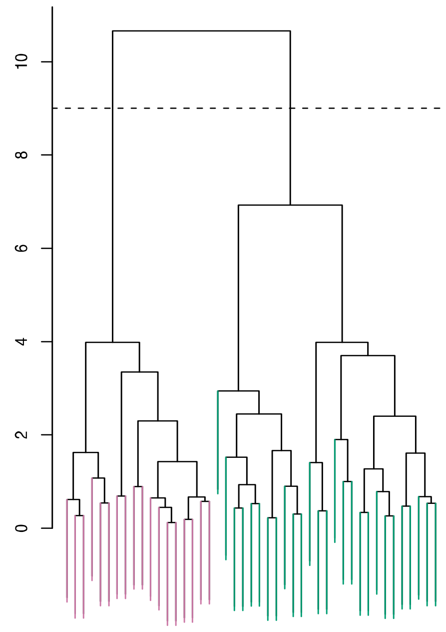
Step 9: {1,2,3,4,5,6,7,8,9}

Choosing Clusters

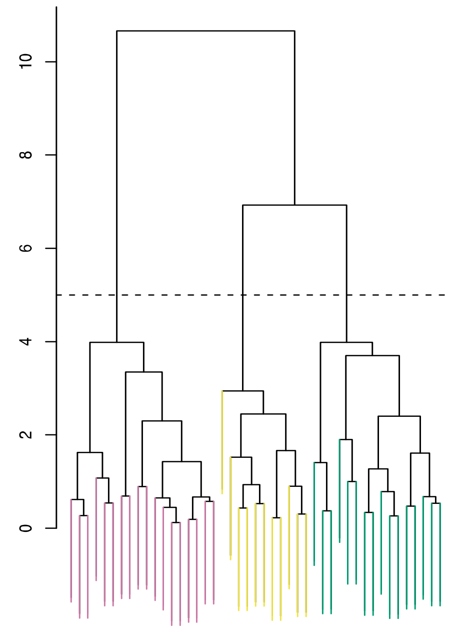
- To choose clusters we draw lines across the dendrogram
- We can form any number of clusters depending on where we draw the break point.



One Cluster



Two Clusters

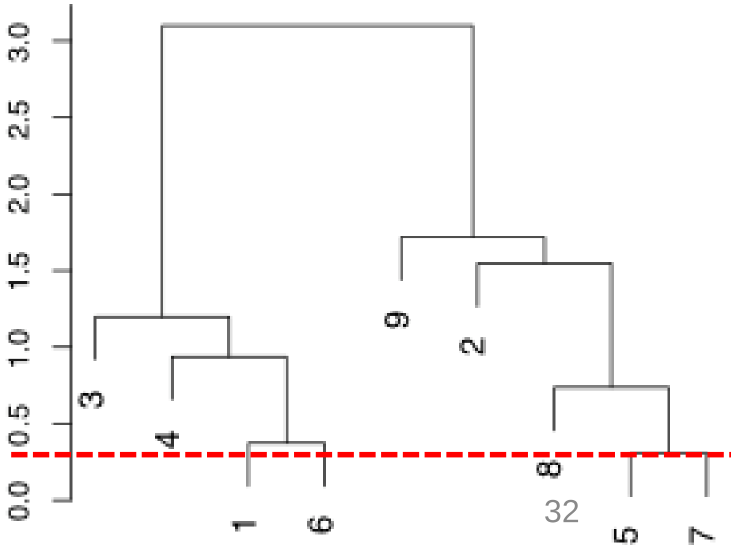


Three Clusters

Dendrogram Interpretation

- Dendrogram: convenient graphic to display a hierarchical sequence of clustering assignments. This is simply a tree where:
  - Each node represents a group
  - Each leaf node is a single observation
  - Root node is the group containing the whole data set
  - Each internal node has two children, representing the groups that were merged to form it.

- The earlier (lower in the tree) two observations fuse, the more similar they are to each other.
- Observations that fuse later are quite different.
- Height of fusing/merging (on vertical axis) indicates how dissimilar the two groups are

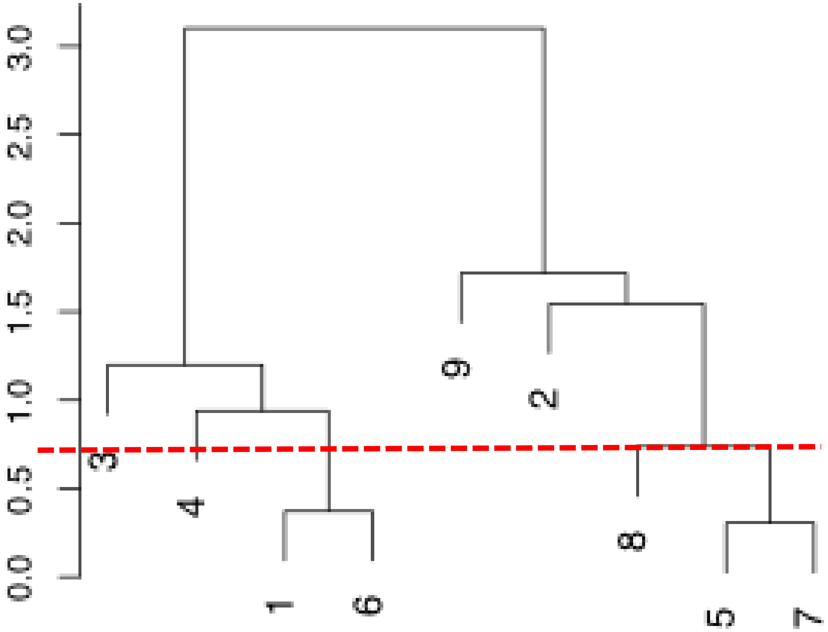


How do we define dissimilarity?

- Implementing hierarchical clustering involves one obvious issue

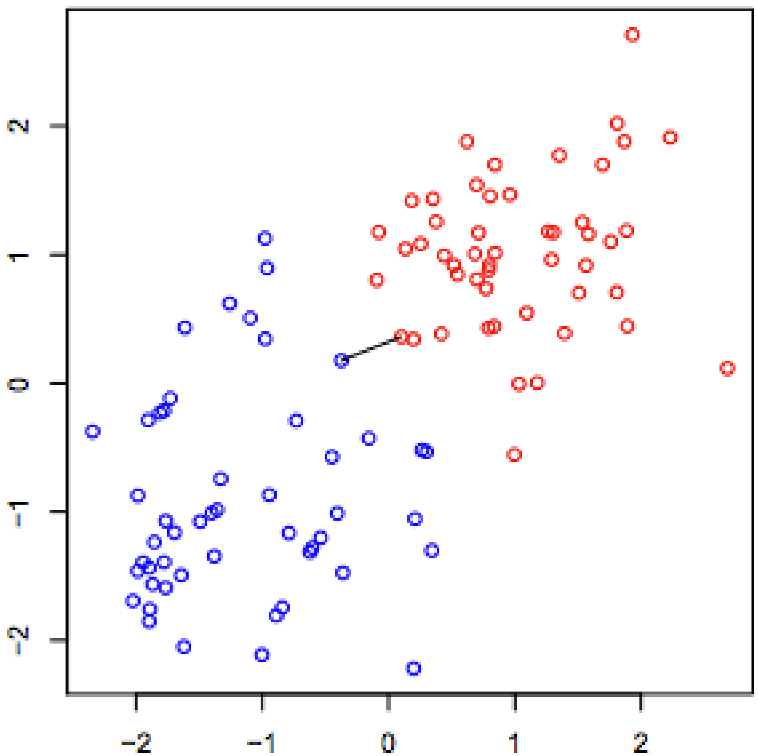
- How do we define the dissimilarity, or linkage, between the fused (5,7,8) cluster and (1,6)?

- There are four options:
  - Single Linkage
  - Complete Linkage
  - Average Linkage
  - Centroid Linkage



Single Linkage

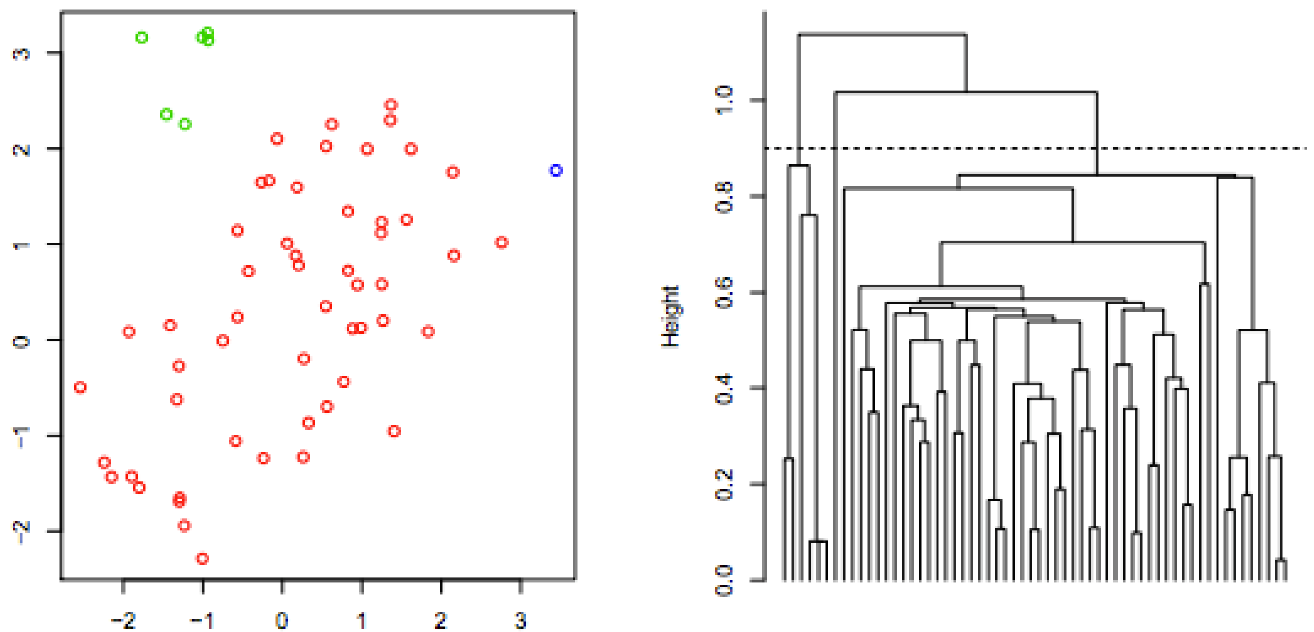
- In single linkage (i.e., nearest-neighbor linkage), the dissimilarity between two clusters is the smallest dissimilarity between two observations in opposite clusters.



Single linkage score is the distance of the closest pair

Single Linkage Example

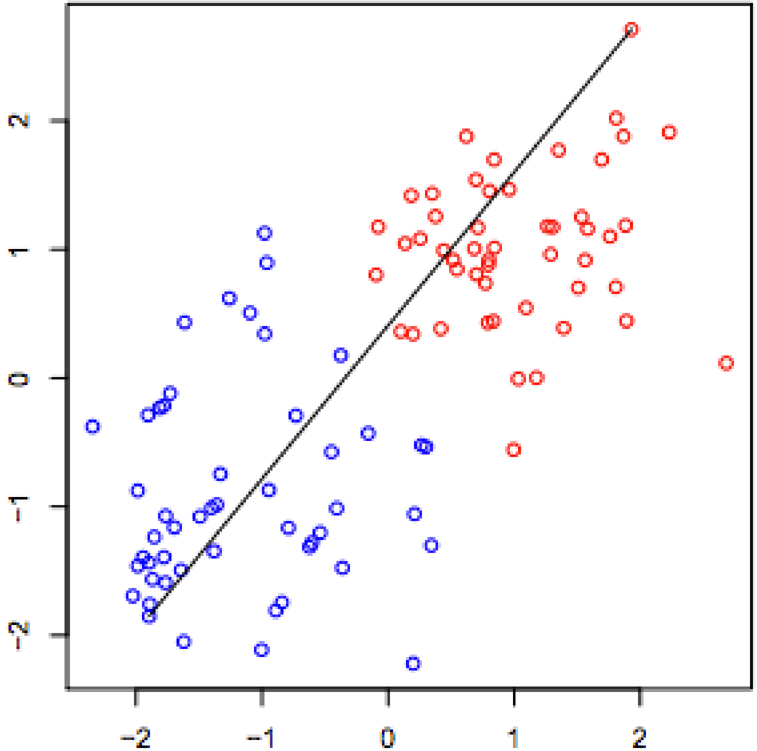
- Here n = 60 (n: number of obs.). Cutting the tree at height=0.9 gives the following clustering assignments marked by colours



- Cut interpretation: for each observation obs_i, there is another observation obs_j in its cluster and their distance is <=0.9

Complete Linkage

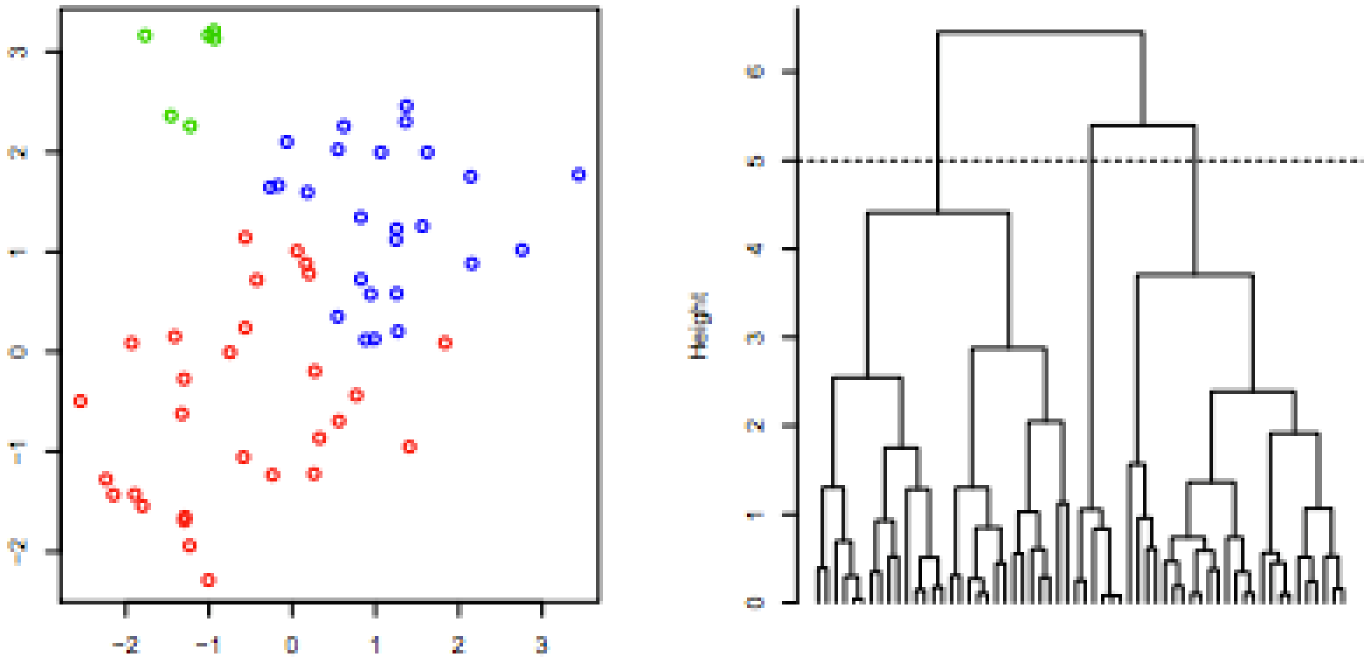
- In complete linkage (i.e., furthest-neighbor linkage), dissimilarity between two clusters is the largest dissimilarity between two observations in opposite clusters



Complete linkage score is the distance of the furthest pair

Complete Linkage Example

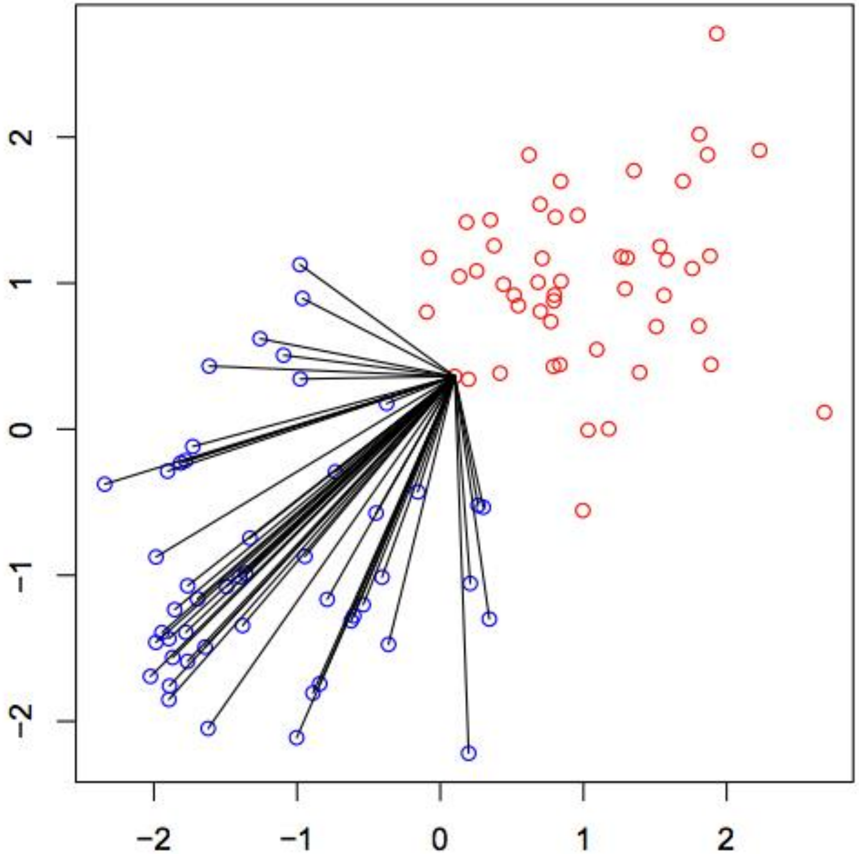
- Same data as before. Cutting the tree at height 5 gives the clustering assignments marked by colours



- Cut interpretation: for each observation obs_i, every observation obs_j in its cluster satisfies that their distance is <=5

Average Linkage

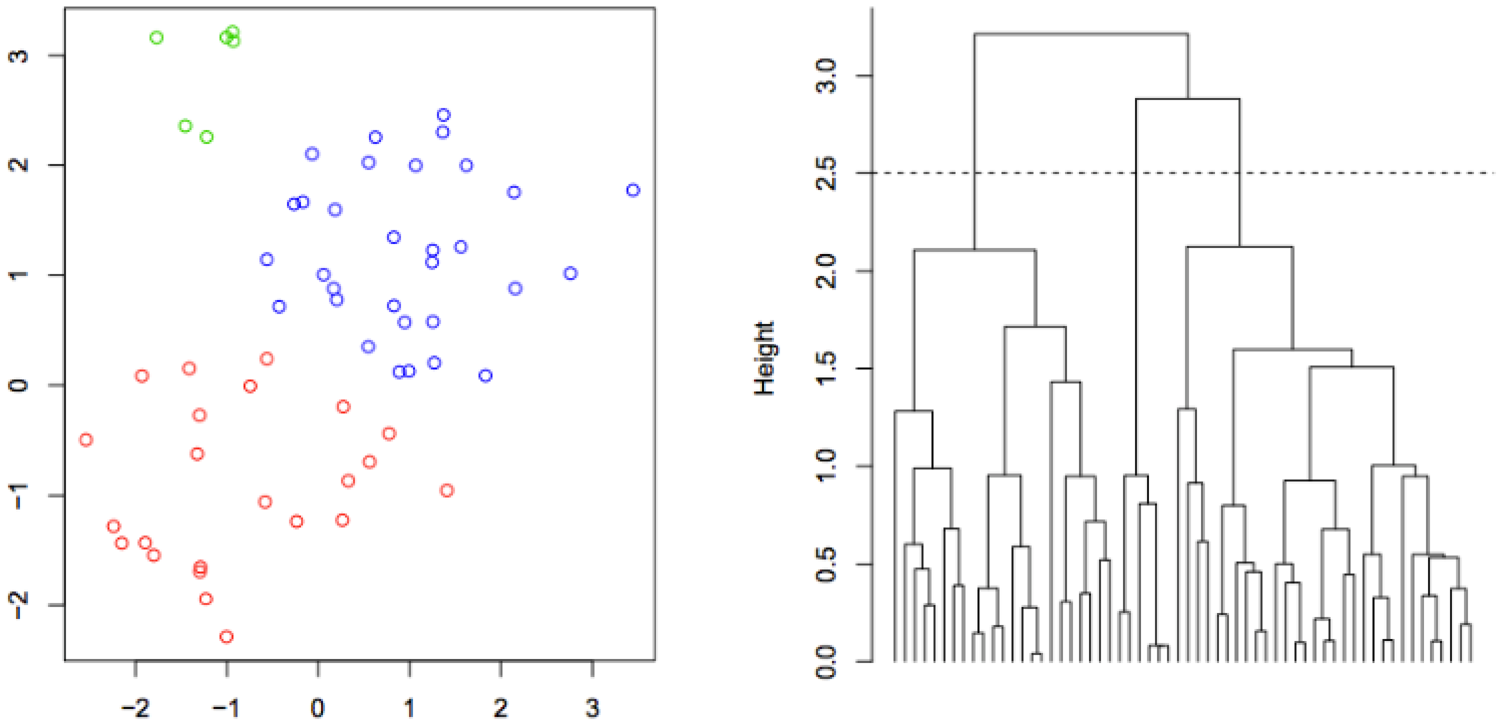
- In average linkage, the dissimilarity between two clusters is the average dissimilarity over all points in opposite clusters



Average linkage score is the average distance across all pairs

Average Linkage Example

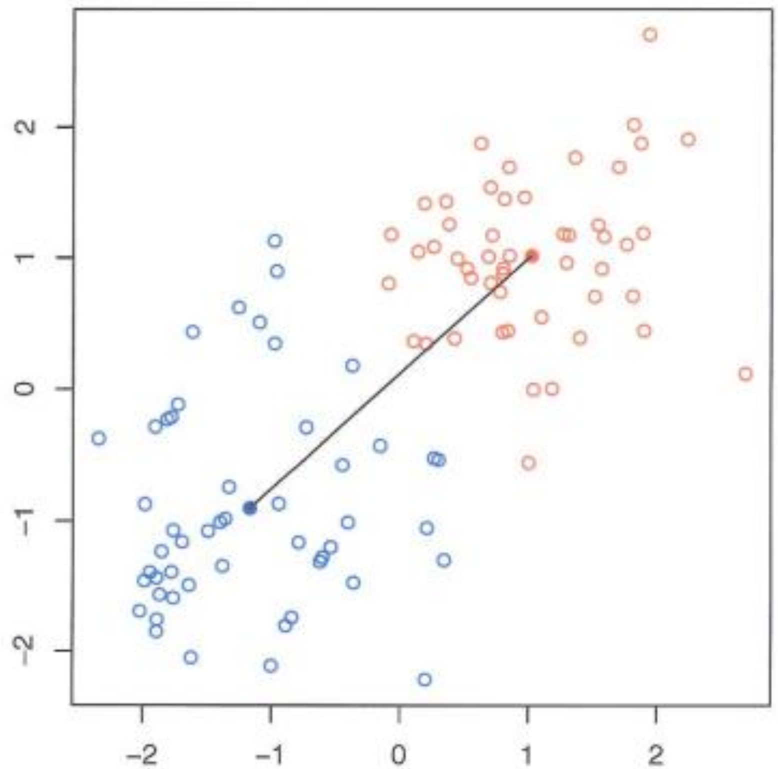
- Same data as before. Cutting the tree at height 2.5 gives clustering assignments marked by the colours



- Cut Interpretation: there really isn't a good one!

Centroid Linkage

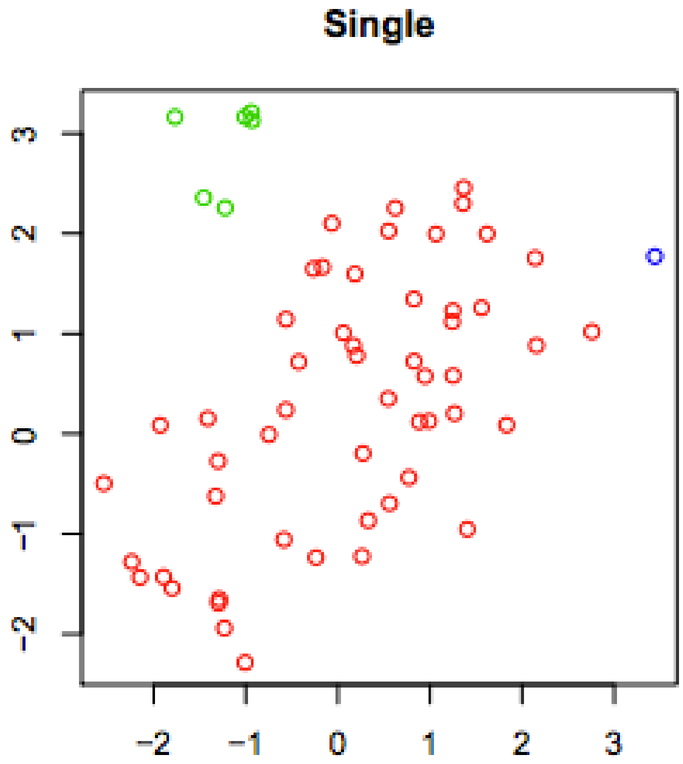
- Centroid linkage score is the distance between the cluster centroids (cluster average)



- Cut Interpretation: There isn't one.

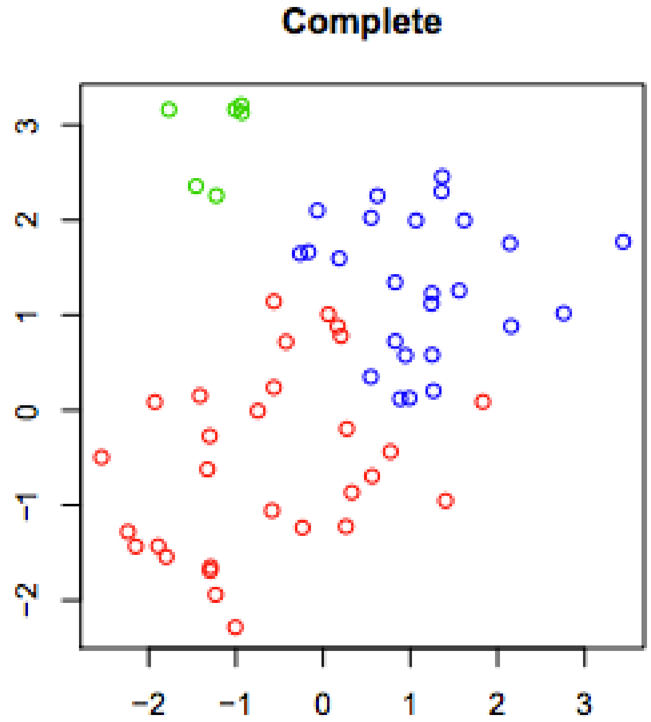
Shortcomings of Single Linkage

- Single linkage can have some practical problems:
  - Single linkage suffers from chaining. In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore clusters can be too spread out, and not compact enough.



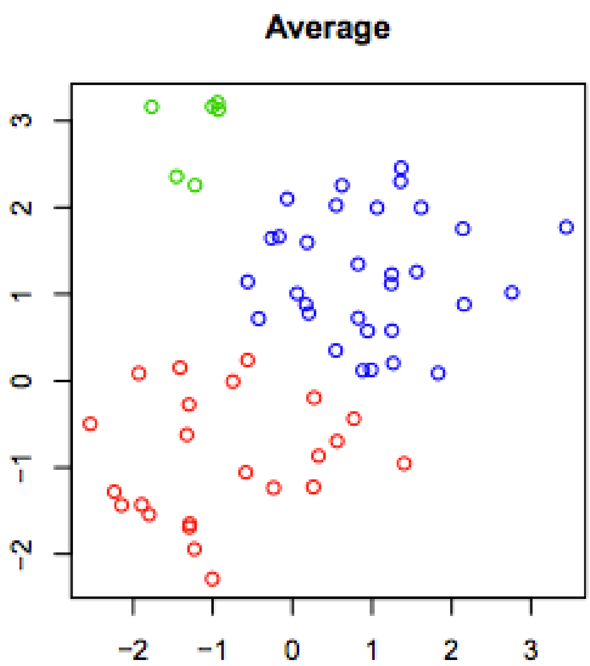
Shortcomings of Complete Linkage

- Complete linkage can have some practical problems:
  - Complete linkage avoids chaining, but suffers from crowding. Because its score is based on the worst-case dissimilarity between pairs, a point can be closer to points in other clusters than to points in its own cluster. Clusters are compact, but not far enough apart.



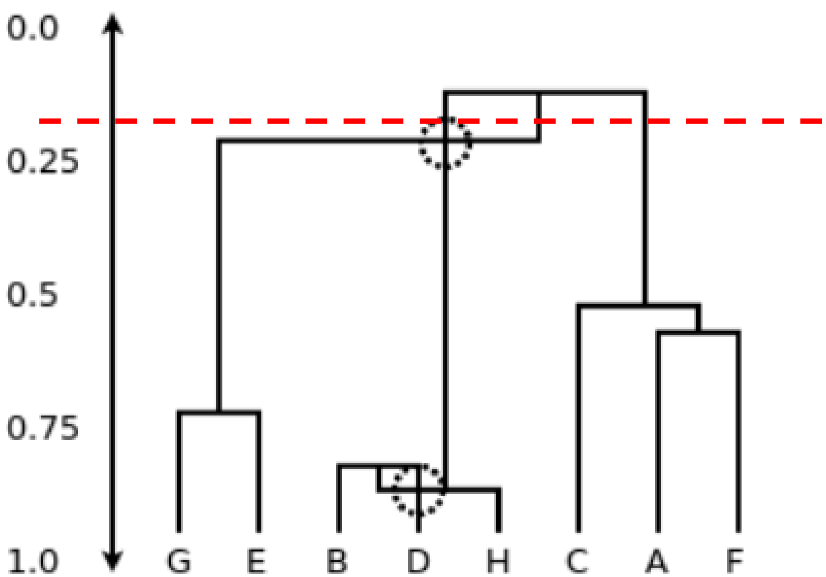
Shortcomings of Average Linkage

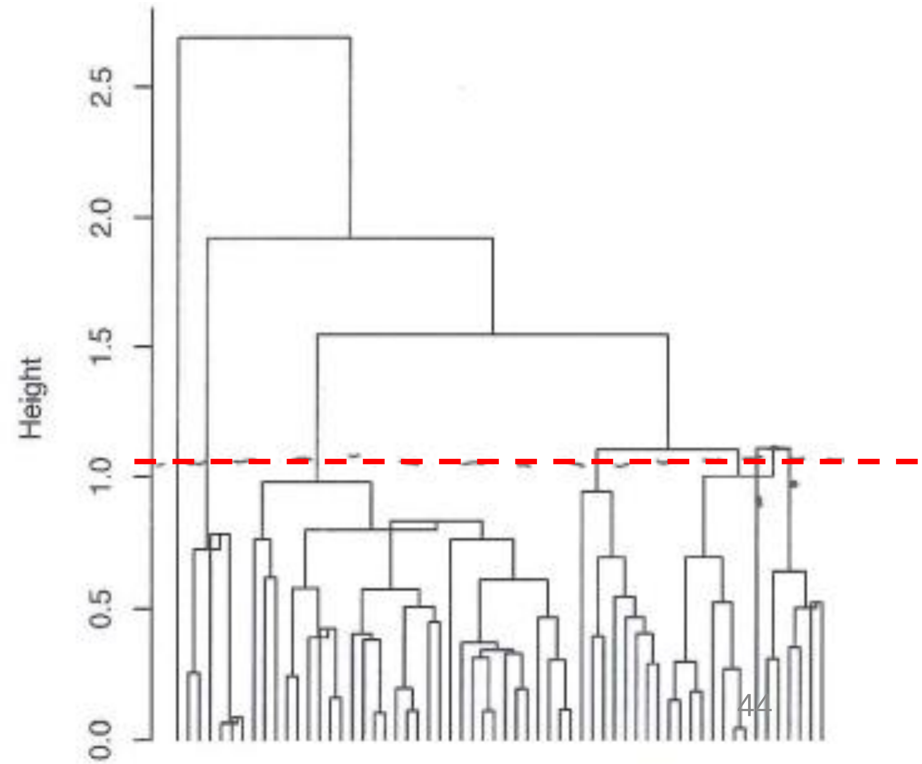
- Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart.
- Average linkage isn't perfect, it has its own problems:
  - It is not clear what properties the resulting clusters have when we cut an average linkage tree at given height h. Single and complete linkage trees each had simple interpretations.



Shortcoming of Centroid Linkage

- Centroid linkage may produce a dendrogram with inversions
  - mess up the visualisation, hard to interpret
- Inversion
  - The height of a parent node is lower than the height of its children



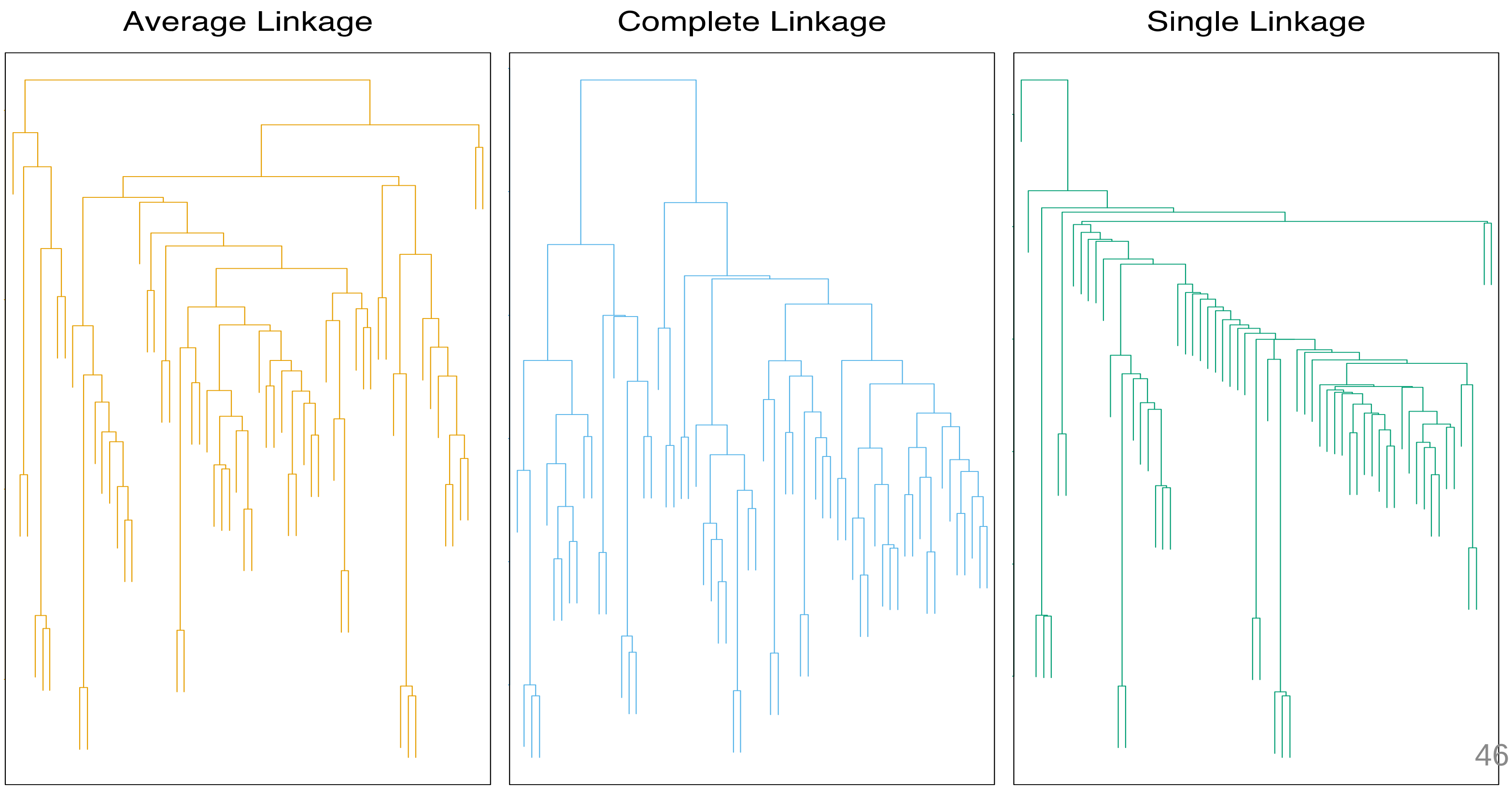


Linkage Can be Important

- Single, complete, average linkage are widely used by statisticians.
  - The dissimilarity scores between merged clusters only increase as we run the algorithm ➔ produces a dendrogram with no inversions
  - Average and complete linkage are generally preferred over single linkage, as they tend to yield evenly sized clusters
  - Single linkage tends to yield extended clusters to which single leaves are fused one by one
- Centroid linkage is often used in biology.
  - It is simple, easy to understand and easy to implement
  - But it may have inversion

Linkage Can be Important

- Here we have three clustering results for the same data
- The only difference is the linkage method but the results are very different



Designing a Clever Radio System

- Suppose we have a bunch of songs, and dissimilarity scores between each pair.
- We are building a clever radio system
  - a user is going to give us an initial song, and
  - a measure of how "risky" s/he is going to be, i.e., maximal tolerable dissimilarity between suggested songs
- The system will recommend songs that a user might like
- How could we use hierarchical clustering, and with what linkage?

Exercise

- Suppose that we have 5 observations, for which we compute a similarity (distance) matrix as follows:

	A	B	C	D	E
A	0				
B	9	0			
C	3	7	0		
D	6	5	9	0	
E	11	10	2	8	0

Try this yourself using the other 2 linkage measures

Centroid linkage is not feasible

- On the basis of the similarity matrix, sketch the dendrogram that results from hierarchically clustering these 5 observations using complete linkage.

Hierarchical Clustering in R

- The hclust() function implements hierarchical clustering in R.
- Task: To plot the hierarchical clustering dendrogram using complete, average and single linkage clustering (data x is from the k-means example)

```
hc.complete <- hclust(dist(x),method="complete")
```

The dist() function is used to compute the 49*49 inter-observation Euclidean distance matrix

```
hc.average <- hclust(dist(x),method="average")
hc.single <- hclust(dist(x),method="single")

par(mfrow=c(1,3))  #show plots in 1 row, 3 columns
plot(hc.complete, main="Complete Linkage",xlab="",ylab="",cex=0.9)
plot(hc.average, main="Average Linkage",xlab="",ylab="",cex=0.9)
plot(hc.single, main="Single Linkage",xlab="",ylab="",cex=0.9)
```

dist() Function

dist() computes the distances between the rows of a data matrix. If a matrix A has n rows, then dist(A) is of size (n-1) × (n-1).

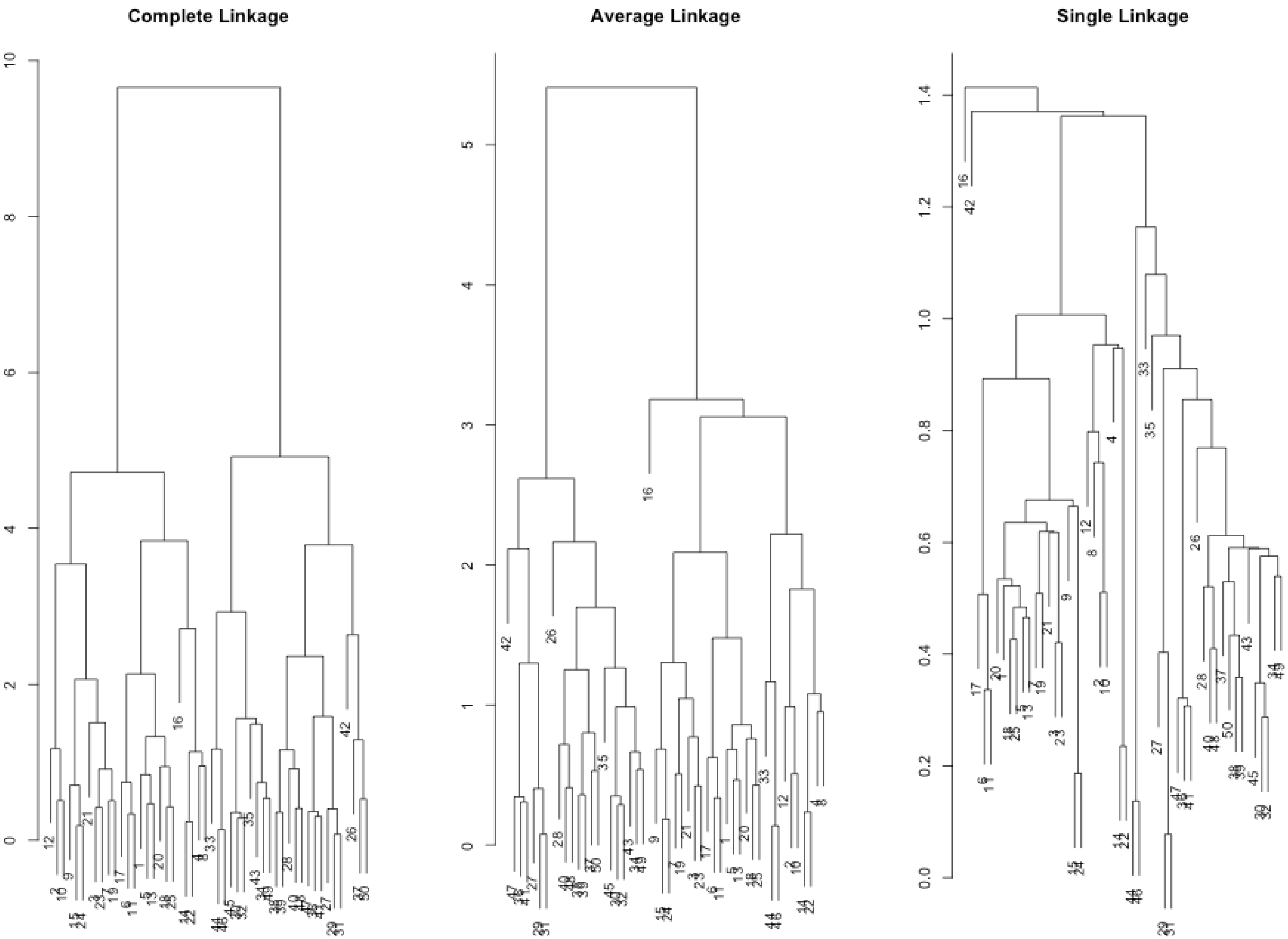
Example 1:

```
> y = matrix(1:6,ncol=2)
> y
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> dist(y)
         1        2
2 1.414214
3 2.828427 1.414214
```

Example 2:

```
> z=matrix(rnorm(5*2),ncol=2)
> dist(z)
          1         2         3         4
2 1.6347493
3 0.4802217 2.0450818
4 2.8124274 1.1983727 3.1887067
5 3.2282552 1.5970885 3.6413656 0.6138656
```

Hierarchical Clustering in R



Hierarchical Clustering in R

- To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the cutree() function

```
cutree(hc.complete,2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
     2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
cutree(hc.average,2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
     2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
cutree(hc.single,2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
     1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

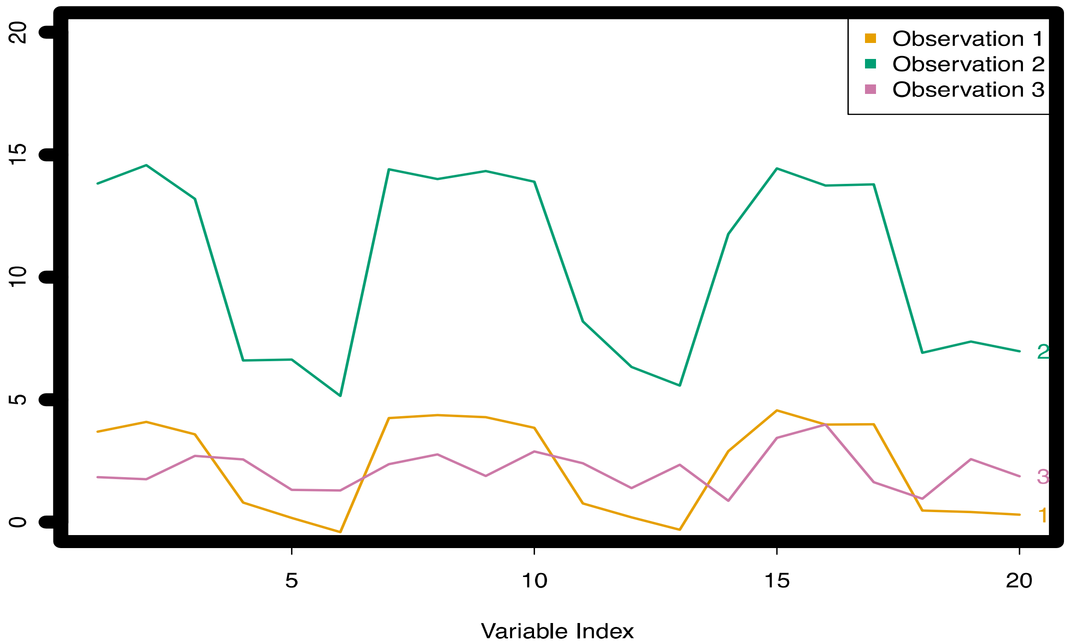
Choice of Dissimilarity Measure

- So far, we have considered using Euclidean distance as the dissimilarity measure

- However, an alternative measure that could make sense in some cases is the correlation based distance
  - It considers two observations to be similar if their features are highly correlated, even though the observed values may be apart in terms of Euclidean distance.

Comparing Dissimilarity Measures

- In this example, we have 3 observations and p = 20 variables
- In terms of Euclidean distance obs. 1 and 3 are similar
- However, obs. 1 and 2 are highly correlated so would be considered similar in terms of correlation measure



The choice of dissimilarity measure is very important, as it has a strong effect on the resulting dendrogram.

It depends on

- What type of data being clustered
- What scientific question at hand

Online Shopping Example

- Suppose we record the number of purchases of each item (columns) for each customer (rows)

- Using Euclidean distance, customers who have purchases very little will be clustered together

- Using correlation measure, customers who tend to purchase the same types of products will be clustered together even if the magnitude of their purchase may be quite different
  - e.g., Shoppers who have bought items A and B but never C or D

Correlation-Based Distance in R

- Suppose we have 4 customers and 3 items to choose from.
  Our dataset (matrix) `purchase` is of the size 4*3.

```
> purchase <- matrix(rnorm(4*3),ncol=3)    #4 observations, 3 features
```

- We are to use the correlation between customers (rows) as distances.
- Problem 1: `cor(purchase)` calculates correlations between columns (size 3*3).
  - Use transpose of the matrix, in R the `t()` function: `t(purchase)` (size 3*4)
  - Then `cor(t(purchase))` calculates the correlation between customers (size 4*4).
- Problem 2: correlations between -1 ... +1 and negative distances make no sense
  - Use `1-cor(t(purchase))` to bring all the values between 0 and 2 (size 4*4).
- Problem 3: need a distance matrix form that `hclust()` recognises
  - `as.dist()` converts an arbitrary square symmetric matrix into a triangular matrix
  - 1-cor(t(purchase)) is 4*4, and as.dist(1-cor(t(purchase))) is 3*3 and is triangular.

```
> dd <- as.dist(1-cor(t(purchase)))
> par(mfrow=c(1,1))    #It will show on the whole of the screen
> plot(hclust(dd, method="complete"), xlab="", ylab="",
      main="Complete Linkage with Correlation-Based Distance")
```

as.dist() and dist()

```
> z=matrix(c(1,9,5,7,6,2,10,12,8,4,3,11),ncol=3)
> z
     [,1] [,2] [,3]
[1,]    1    6    8
[2,]    9    2    4
[3,]    5   10    3
[4,]    7   12   11
> t(z)
     [,1] [,2] [,3] [,4]
[1,]    1    9    5    7
[2,]    6    2   10   12
[3,]    8    4    3   11
> cor(t(z))
            [,1]       [,2]        [,3]       [,4]
[1,]  1.00000000 -0.8461538 -0.03846154  0.8910421
[2,] -0.84615385  1.0000000 -0.50000000 -0.9958706
[3,] -0.03846154 -0.5000000  1.00000000  0.4193139
[4,]  0.89104211 -0.9958706  0.41931393  1.0000000
```

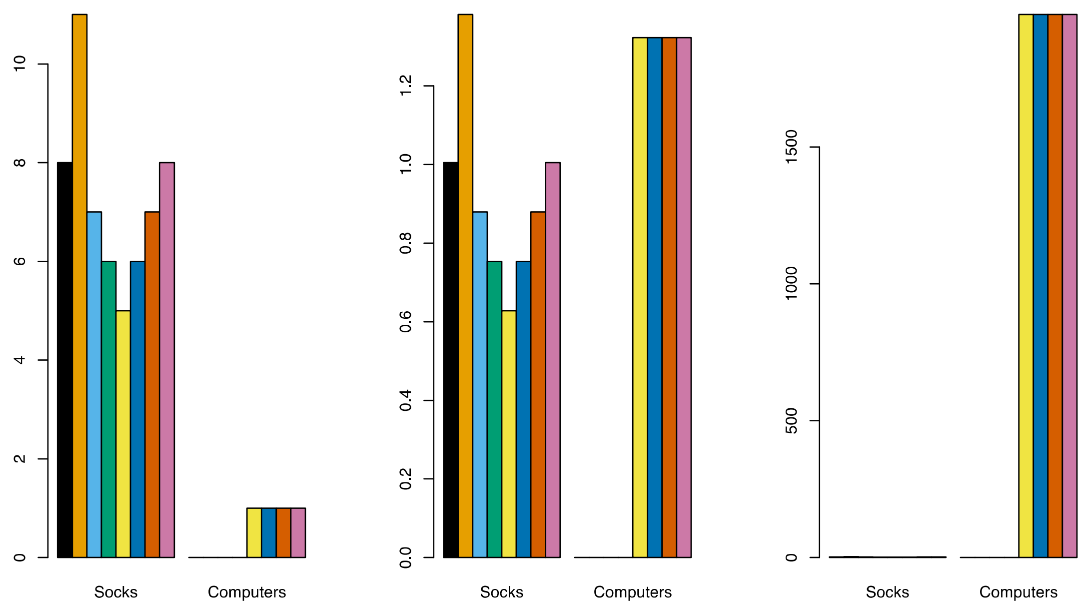
- dist() will calculate the distance matrix between rows of the matrix
- as.dist() only will try to coerce an object to a distance matrix

```
> u=1-cor(t(z))
> u
          [,1]     [,2]      [,3]      [,4]
[1,] 0.0000000 1.846154 1.0384615 0.1089579
[2,] 1.8461538 0.000000 1.5000000 1.9958706
[3,] 1.0384615 1.500000 0.0000000 0.5806861
[4,] 0.1089579 1.995871 0.5806861 0.0000000
> dist(u)
          1         2         3
2 3.2542319
3 1.5808715 2.6749045
4 0.5056848 3.4394748 1.3357609
> as.dist(u)
          1         2         3
2 1.8461538
3 1.0384615 1.5000000
4 0.1089579 1.9958706 0.5806861
```

cor(matrix): return a symmetric correlation matrix between columns of matrix

Standardising the Variables

- Consider an online shop that sells two items: socks and computers
  - Left: In terms of quantity, socks have higher weight
  - Center: After standardising, socks and computers have equal weight
  - Right: In terms of pound sales, computers have higher weight



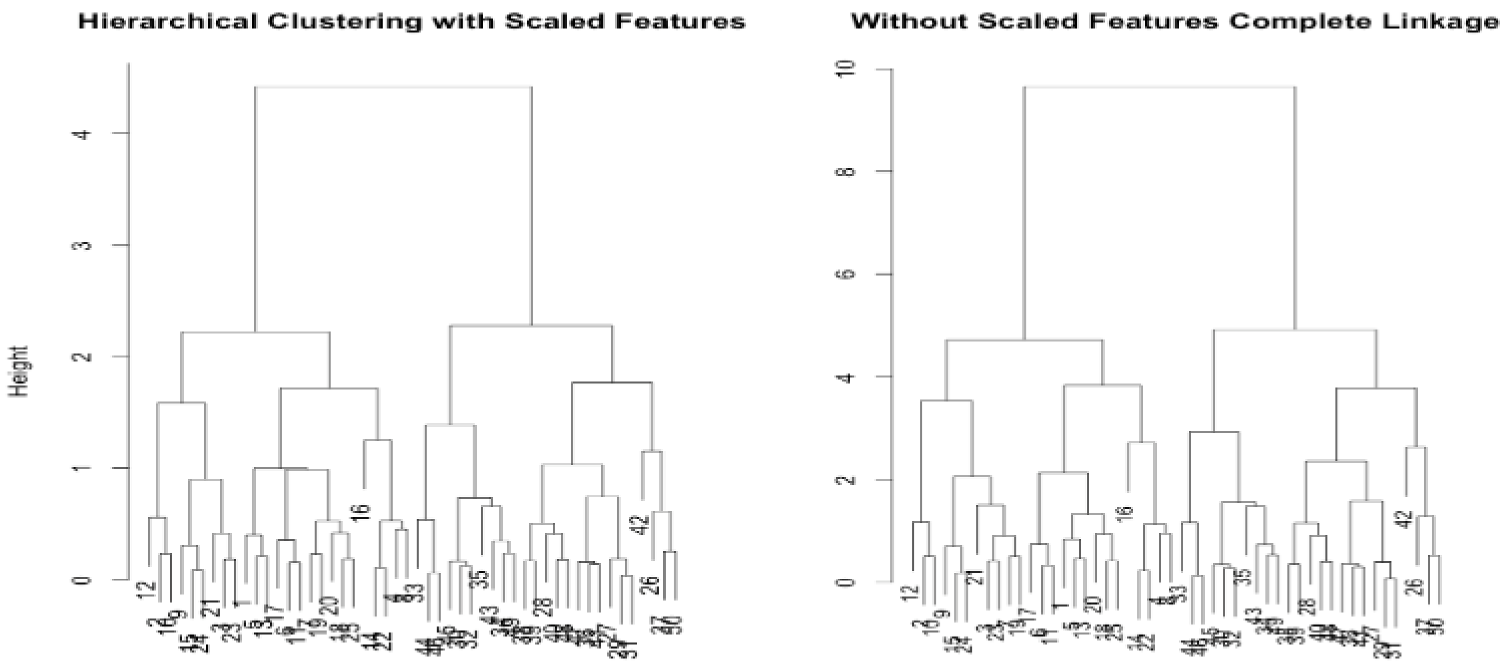
Each shopper shown in a different colour

Scaling the Variables

- If the variables are measured on different scales, we might want to scale the variables to have standard deviation one.
  - In this way each variable will in effect be given equal importance in the hierarchical/K-means clustering performed.
  - This actually applies to many statistical learning methods as a pre-processing step.
- Whether or not to scale the variables before computing the dissimilarity measure depends on the application at hand.

Scaling in R

```
xsc <- scale(x)    # Used to scale the variables before performing HC
par(mfrow=c(1,2))
plot(hclust(dist(xsc), method="complete"),
     main="Hierarchical Clustering with Scaled Features")
plot(hc.complete, xlab="", ylab="", cex=0.9,
     main="Without Scaled Features Complete Linkage")
```



Outline

- What is Clustering?
- K-Means Clustering
- Hierarchical Clustering
- Final Thoughts

Practical Issues in Clustering

- In order to perform clustering, some decisions must be made:
  - Should the features first be standardised? i.e. Have the variables centered to have a mean of zero and standard deviation of one?
  - What dissimilarity measure should be used?
  - In case of hierarchical clustering:
    - What type of linkage should be used?
    - Where should we cut the dendrogram in order to obtain clusters?
  - In case of K-means clustering:
    - How many clusters should we look for the data?

  ➔ These decisions will lead to very different results
- In practice, we try several different choices, and look for the one with the most useful or interpretable solution. There is no single right answer!

Final Thoughts

- Most importantly, one must be careful about how the results of a clustering analysis are reported

- These results should not be taken as the absolute truth about a dataset

- Rather, they should constitute a starting point for the developments of a scientific hypothesis and further study, preferably on independent data