Clustering

Torben Tvedebrink

August 23

Introduction

Terminology

- Supervised learning (labeled training data)
    - Classification
    - Regression
- Unsupervised learning (describe hidden structure from “unlabeled” data)
    - PCA
    - Clustering (K-means, …)

Unsupervised learning

- Unsupervised learning is more subjective; no simple goal of analysis (e.g. prediction of a response)
- Important
    - subgroups of breast cancer patients grouped by their gene expression measurements
    - groups of shoppers characterized by their browsing and purchase histories
    - movies grouped by the ratings assigned by movie viewers
- Unravel hidden/non-obvious structure
- Easier to obtain unlabeled data than labeled data

Clustering

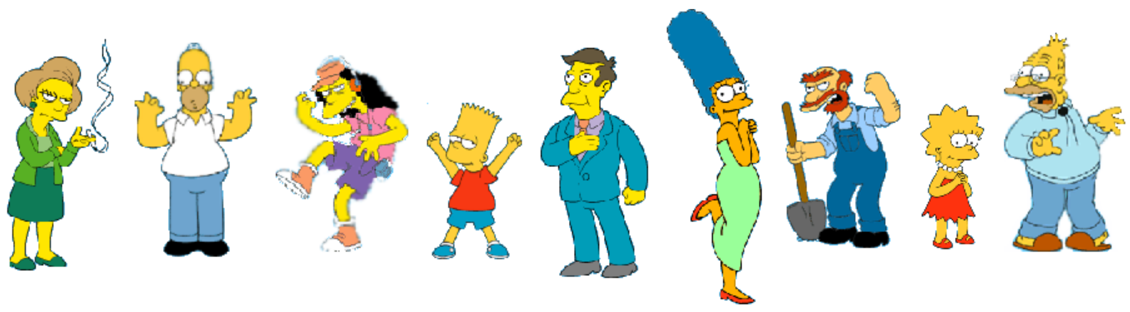
PCA vs clustering

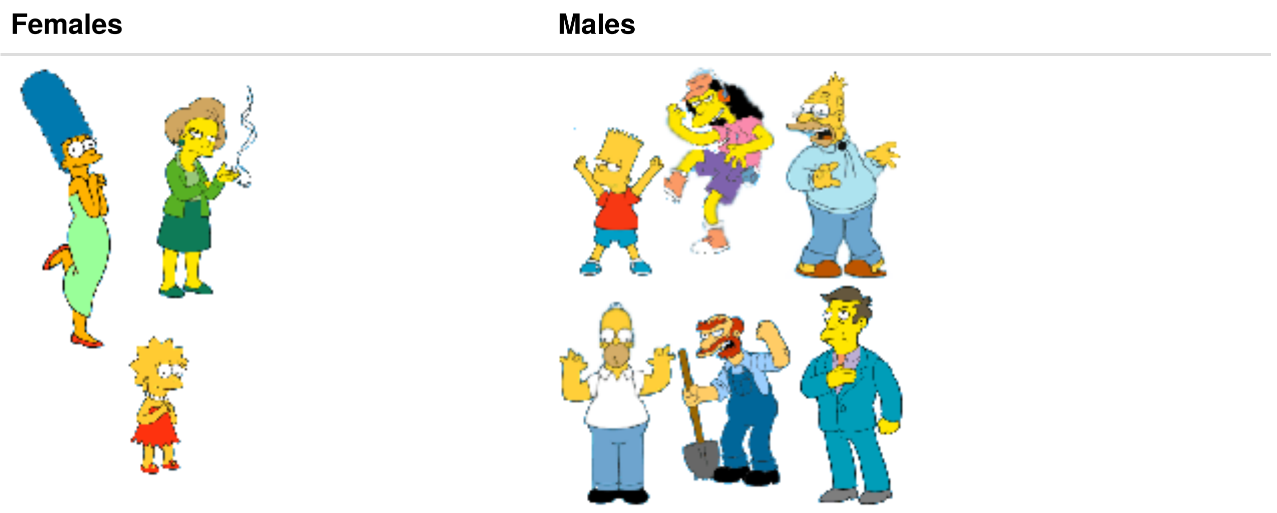
- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance
- Clustering looks for homogeneous subgroups among the observations (ie. the grand dataset in heterogeneous)

Clustering for market segmentation

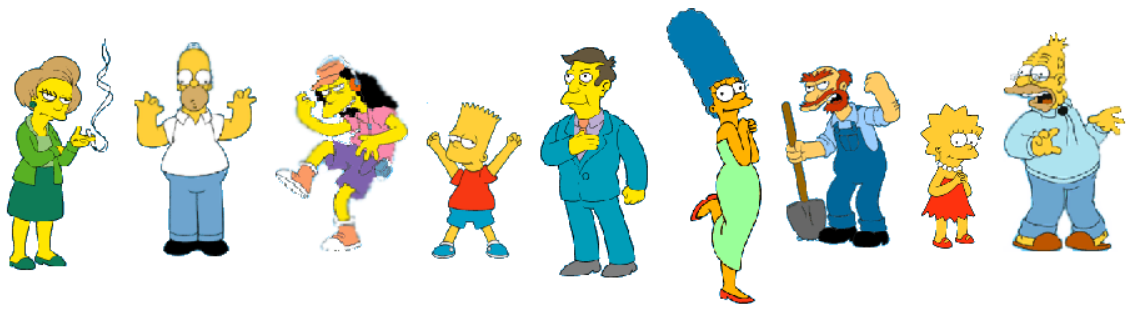
- Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.
- Our goal is to perform market segmentation by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.
- The task of performing market segmentation amounts to clustering the people in the data set.

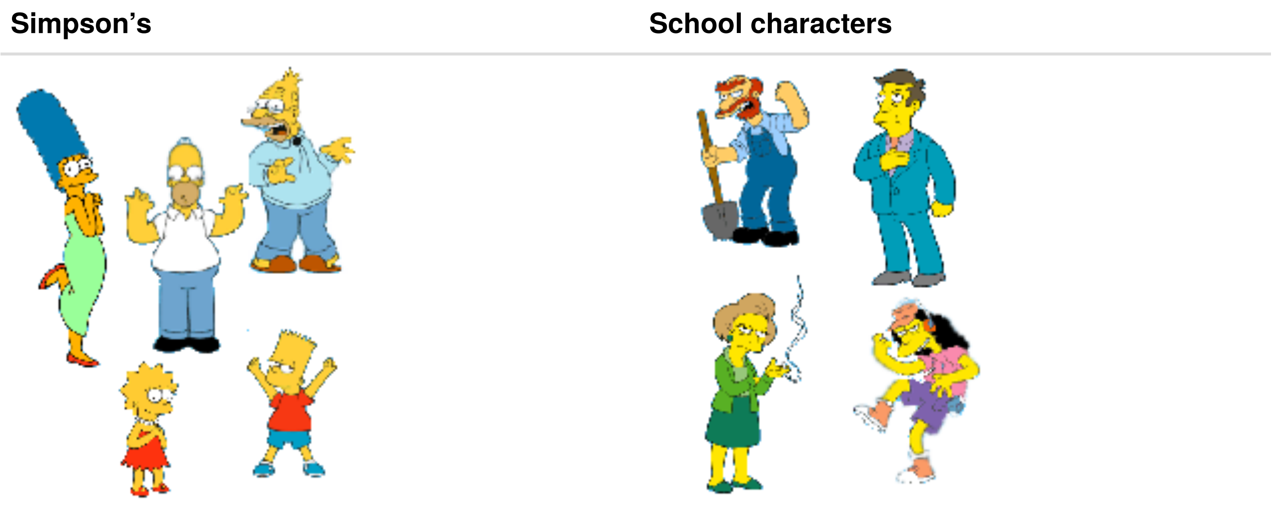
Clustering is subjective





Clustering is subjective





Two clustering methods

- Partitioning clustering:
    - K-means clustering: groups the observations into a pre-specified number of clusters.
- Hierarchical clustering: Number of clusters not known in advance.
    - Result is a tree-like visual representation of the observations (a dendrogram)
    - Allows us to view at once the clustering obtained for each possible number of clusters, from 1 to n

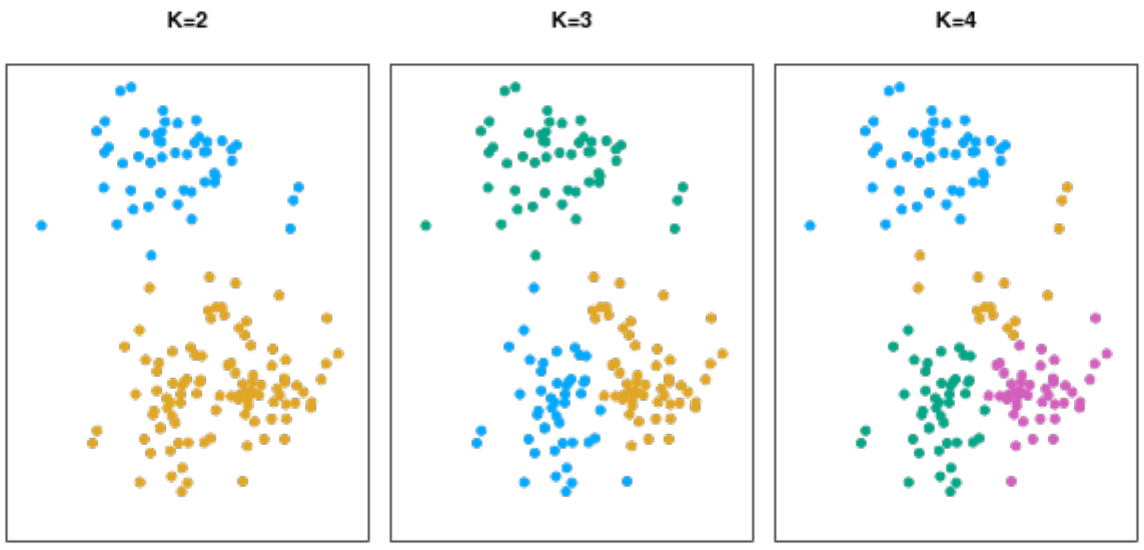
K-means

K-means only works for numerical data. Hence, if our data contains non-numerical columns, we either need to discard these, or represent them by numerical values. However, the latter may be dubious, as we introduce a *distance* between categories.

E.g. if the variable `colour` in a dataset contains three *levels*: `green`, `red` and `yellow`, assigning numerical values to these, e.g. `green = 1`, `red = 2` and `yellow = 3` we suddenly impose that there is twice the distance between `green` and `yellow` than between `red` and the other two levels. This may have unwanted side-effects.

K-means

Here is the same data analysed with K-means for $K = 2$, $K = 3$ and $K = 4$.



- K is number of clusters
- Color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm
- No ordering of clusters (color is arbitrary)

A partitioning

- K is given before analysis starts
- Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These

sets satisfy two properties:

1. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_k' = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

$C_1, \ldots, C_K$ is a partitioning of $\{1, \ldots, n\}$.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$.

Idea behind K-means

- The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation (WCV) is as small as possible.
- The within-cluster variation for cluster C_k is a measure $\mathrm{WCV}(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\mathrm{minimize}_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \mathrm{WCV}(C_k) \right\}.$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Within-cluster variation

- Typically we use Euclidean distance

$$\mathrm{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

where $|C_k|$ denotes the number of observations in the kth cluster
- Hence, the problem is

$$\mathrm{minimize}_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

- Can use other distances (for $\mathrm{WCV}(C_k)$) than the Euclidean

K-means clustering algorithm

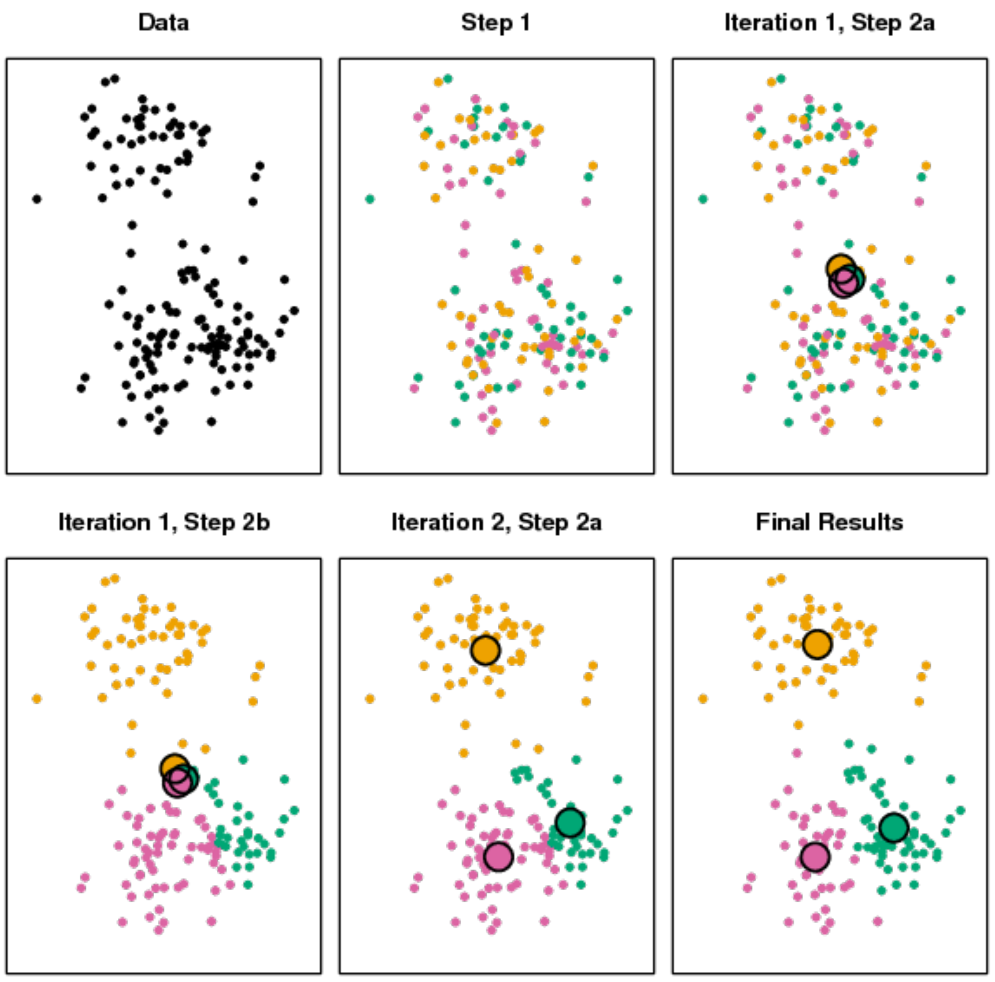
1. Randomly assign a number, from 1 to K, to each of the observations (initial cluster assignments for the observations)
2. Iterate until the cluster assignments stop changing:
   - For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   - Assign each observation to the cluster whose centroid is closest (where closest is defined

using Euclidean distance).

Properties of the algorithm

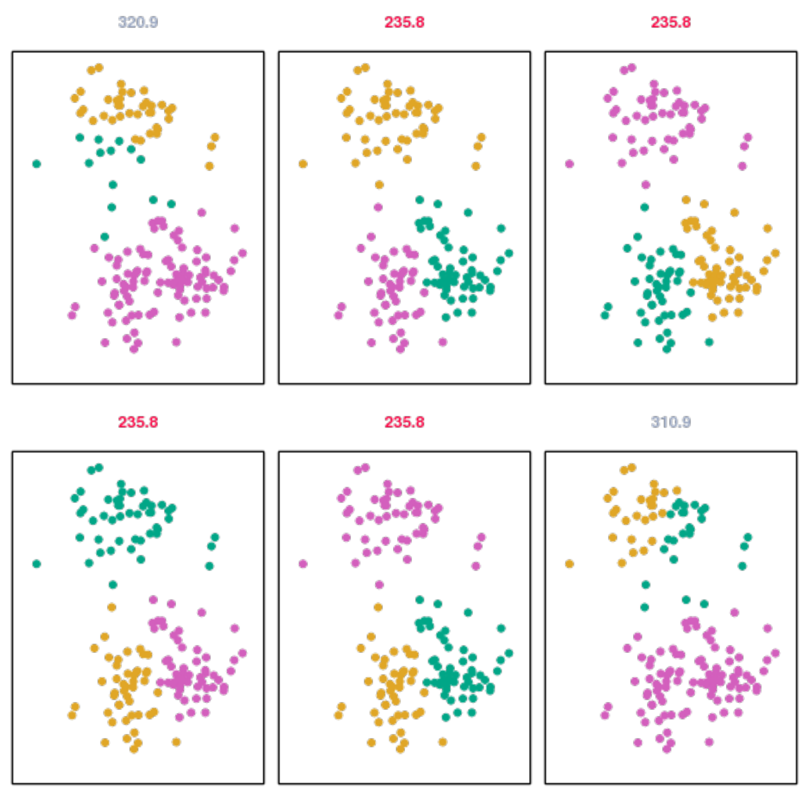
- This algorithm is guaranteed to decrease the value of the objective (minimising WCV) at each step.
- However it is not guaranteed to give the global minimum.

Example



Example

Using different starting values



In R

Use `kmeans`, see `?kmeans`.

Important arguments:

- `x`: the data (as `matrix`, `data.frame` or `tibble`)
- `centers`: if numerical this is K, if matrix the starting centers to be used
- `nstart`: The number of re-runs of the algorithm to decrease the effect of unfortunate initial centers (set `nstart` to `10` or `50`)

Sums of squares

$$\bar{x} = n^{-1} \sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{x}_k = |C_k|^{-1} \sum_{i \in C_k} x_i$$

Total sum of squares and within sums of squares for cluster k:

$$SS_{TOT} = \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad \text{and} \quad SS_{W_k} = \sum_{i \in C_k} (x_i - \bar{x}_k)^2$$

Within sums of squares and between sums of squares:

$$SS_W = \sum_{k=1}^{K} SS_{W_k} \quad \text{and} \quad SS_B = SS_{TOT} - SS_W$$

SS_W can be made arbitrarily small and SS_B increases with K.

Choosing K

- CH Index: Calinski and Harabasz (1974): "A dendrite method for cluster analysis".

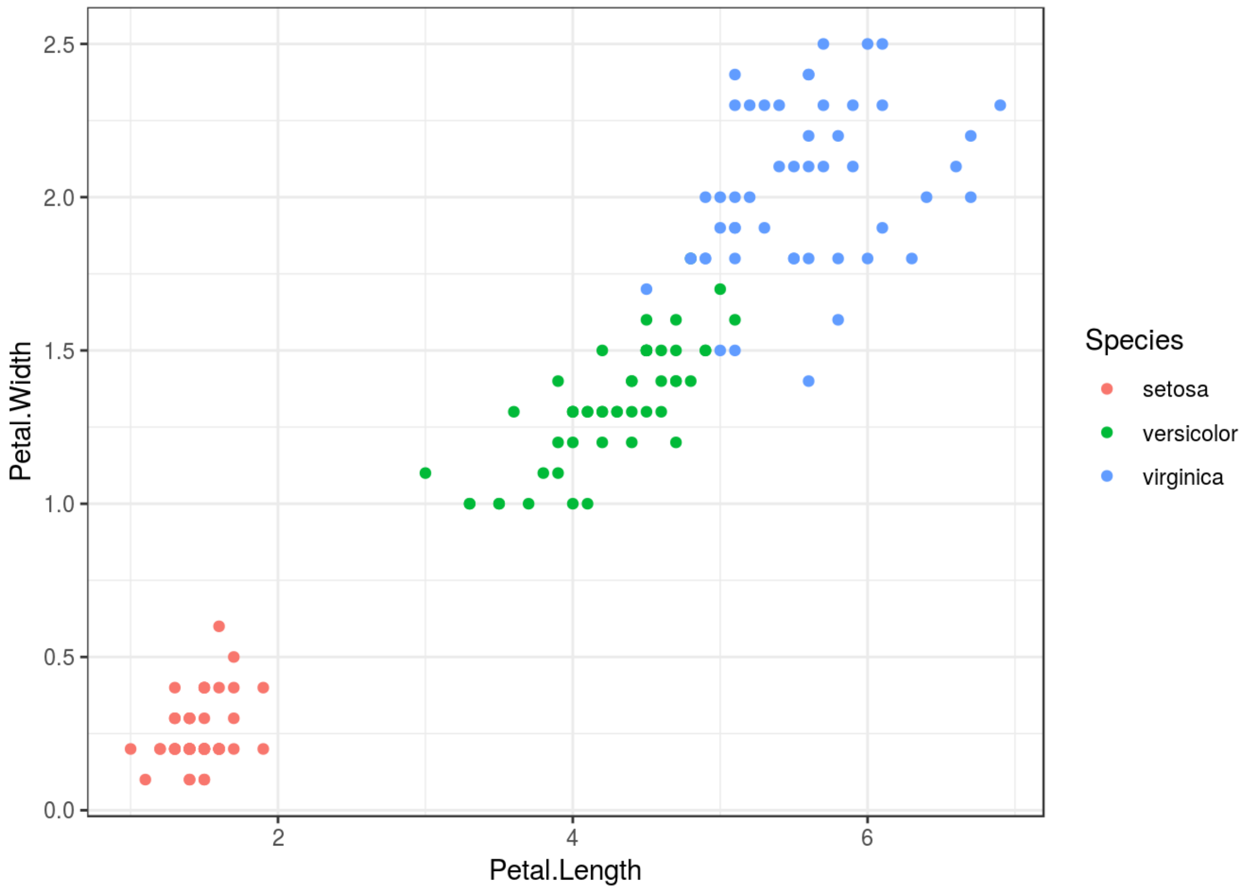
$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)} = \frac{SS_B/(K-1)}{SS_W/(n-K)}$$

  where $B(K)$ and $W(K)$ are between-sum-of-squares and within-sum-of-squares, respectively, for K clusters.

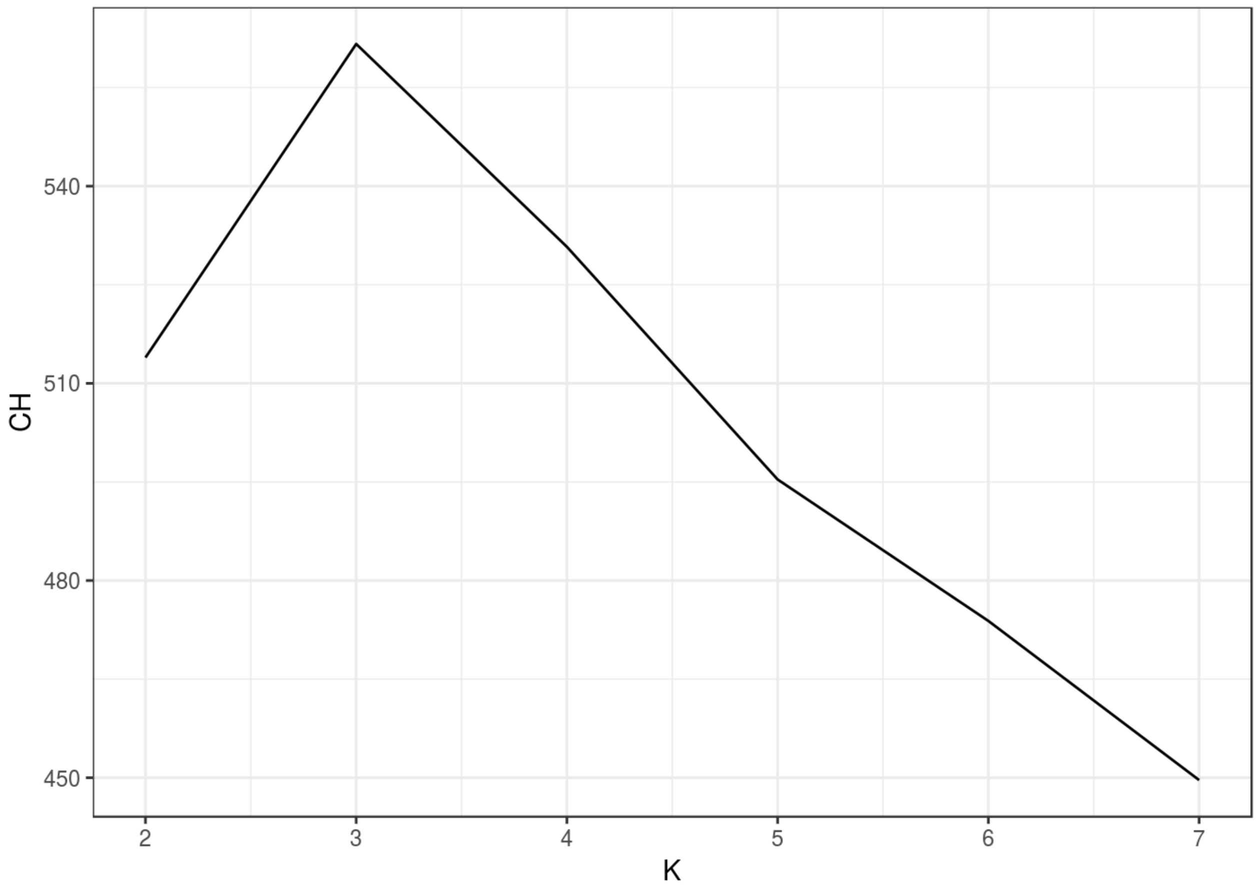
- As large as possible
- This is similar to looking for the "albow" effect in a plot of the drop in SS_W

Example (iris)

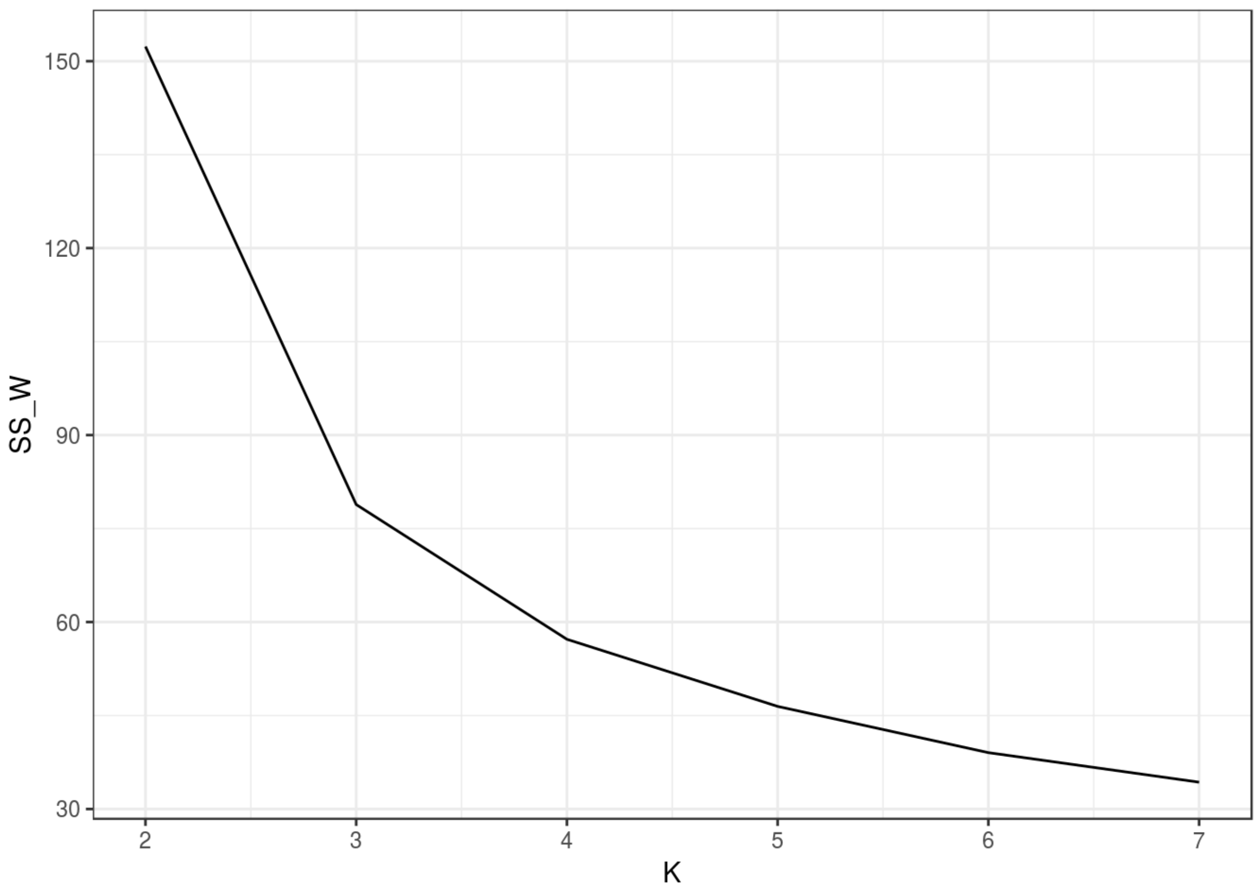
```
iris_noclass <- iris %>% as_tibble() %>% select(-Species)

iris %>% # _noclass %>%
  ggplot(aes(x = Petal.Length, y = Petal.Width)) +
  geom_point(aes(colour = Species))
```



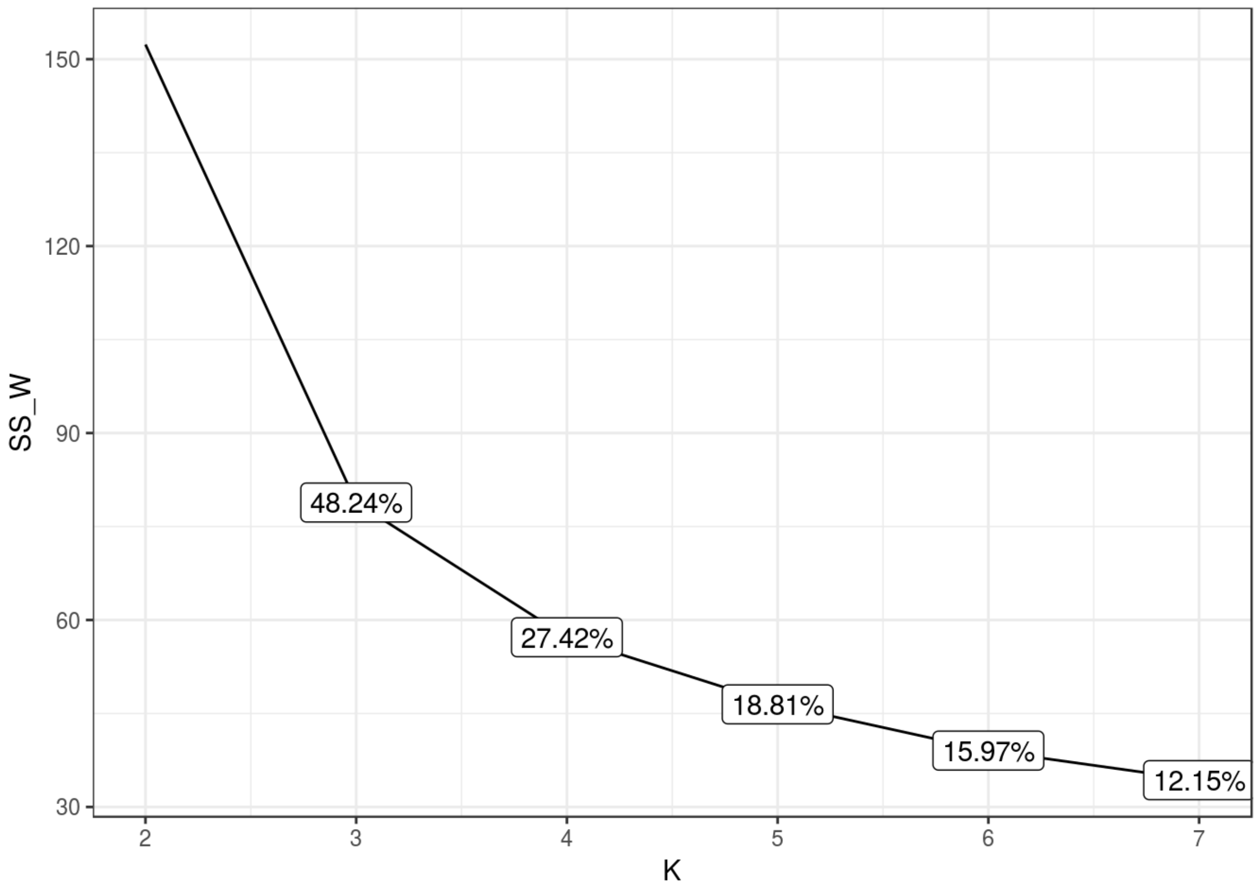
```
i2 <- iris_noclass %>% kmeans(2, nstart = 10)
i3 <- iris_noclass %>% kmeans(3, nstart = 10)
i4 <- iris_noclass %>% kmeans(4, nstart = 10)
i5 <- iris_noclass %>% kmeans(5, nstart = 10)
i6 <- iris_noclass %>% kmeans(6, nstart = 10)
i7 <- iris_noclass %>% kmeans(7, nstart = 10)
iris_kmeans <- tibble(K = 2:7,
                      n = nrow(iris_noclass),
                      SS_B = c(i2$betweenss, i3$betweenss, i4$betweenss, i5$
betweenss, i6$betweenss, i7$betweenss),
                      SS_W = c(i2$tot.withinss, i3$tot.withinss, i4$tot.with
inss, i5$tot.withinss, i6$tot.withinss, i7$tot.withinss),
                      CH = (SS_B/(K-1))/(SS_W/(n-K)),
                      SS_ratio = SS_B/(SS_B+SS_W),
                      SS_W_drop = c(NA,(SS_W[-length(SS_W)] - SS_W[-1])/SS_W
[-length(SS_W)])
                      )

ggplot(iris_kmeans, aes(x = K, y = CH)) + geom_line()
```



```
(iris_drop <- ggplot(iris_kmeans, aes(x = K, y = SS_W)) + geom_line())
```



```
iris_drop + geom_label(data = iris_kmeans %>% filter(K>2), aes(label = paste
0(round(SS_W_drop*100,2), "%")))
```



Alternatives

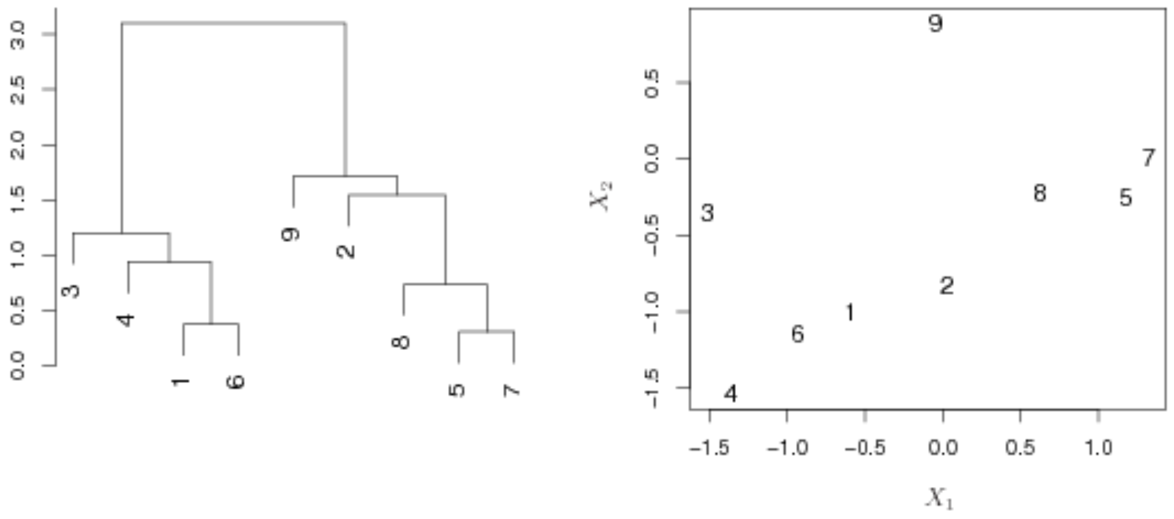
- Instead of the mean as centers, the mediods ("medians") can be used: K-mediods, via `pam` in the `cluster` package
- Soft clustering (model based clustering) giving probabilities of belonging to each cluster, e.g. `library(mclust); vignette("mclust")`
- DBSCAN – Density-Based Spatial Clustering of Applications with Noise, e.g. `library(dbscan)`

Hierarchical clustering

Hierarchical clustering

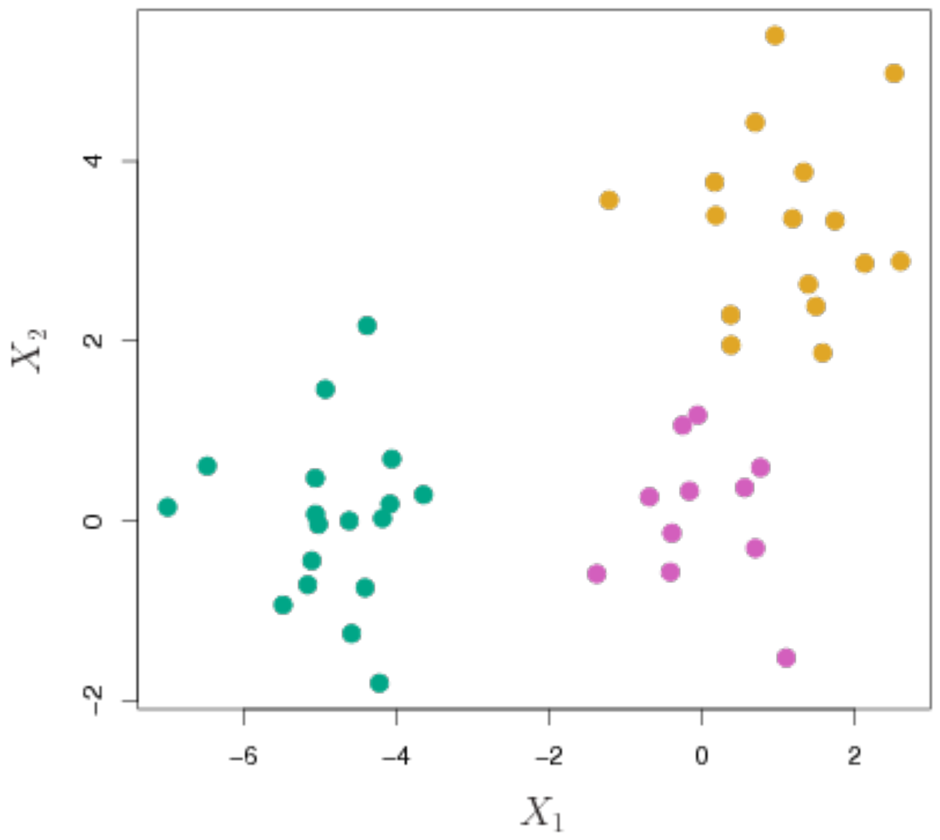
- K-means clustering: pre-specify the number of clusters, K
- Hierarchical clustering is an alternative bottom-up (or agglomerative) approach which does not require that we commit to a particular choice of K.
- Bottom-up: Most common type of hierarchical clustering; a dendrogram is built starting from the leaves and combining clusters up to the trunk.

Example 1

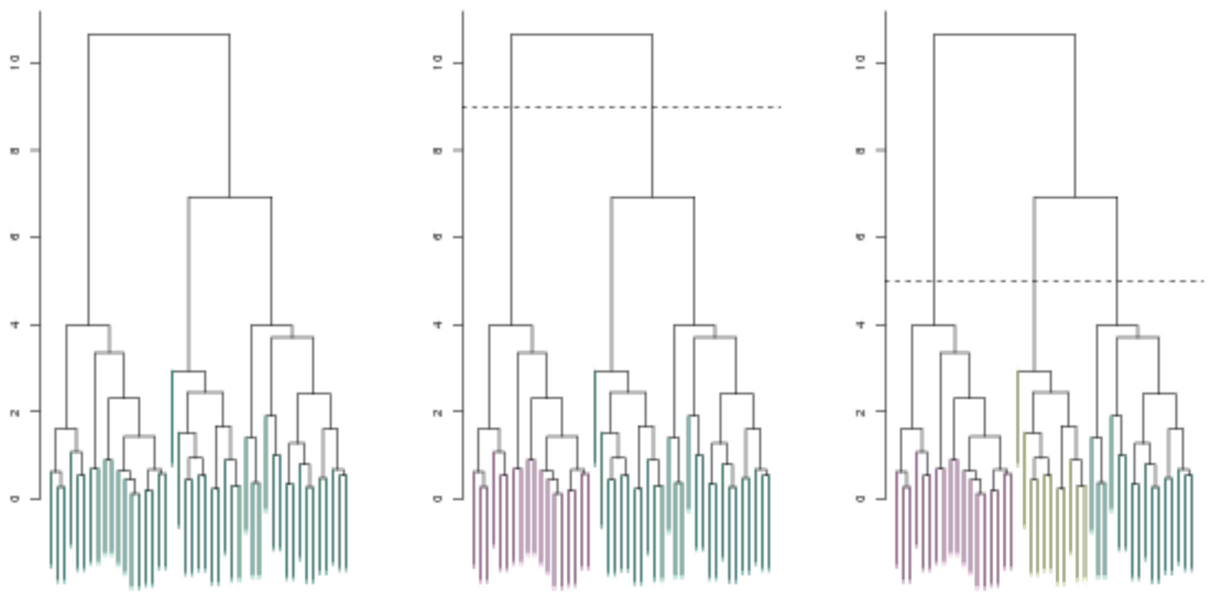


- Obs. 5, 7 are quite similar to each other, as are obs. 1, 6.
- Obs. 9 is no more similar to obs. 2 than it is to obs. 8, 5, and 7, even though obs. 9 and 2 are close together in terms of horizontal distance.
  - Because obs. 2, 8, 5, and 7 all fuse with obs. 9 at the same height, approximately 1.8.

Example 2



Example 2 - Cont’d



What is Dissimilarity

To determine which objects that are alike – and which that are different, we need a (dis)similarity measure.

Let x_i and x_j be the ith and jth observation, respectively. A dissimilarity measure, d, must satisfy:

- $d_{ij} = d(x_i, x_j) = d(x_j, x_i) = d_{ji}$
- $d_{ii} = d(x_i, x_i) = 0$
- $d_{ij} = d(x_i, x_j) = 0$ is and only if $x_i = x_j$.

Pairwise dissimilarities in R

```
(data <- data.frame(x1 = c(0,1,3), x2=c(0,1,4)))
```

```
  x1 x2
1  0  0
2  1  1
3  3  4
```

```
(res <- dist(data, method="euclidian", diag=TRUE, upper=TRUE))
```

```
         1        2        3
1 0.000000 1.414214 5.000000
2 1.414214 0.000000 3.605551
3 5.000000 3.605551 0.000000
```

```
class(res)
```

```
[1] "dist"
```

To enhance the number of dissimilarity measures in R, the `proxy` package can be attached.

```
library(proxy)
summary(pr_DB)
```

```
* Similarity measures:
Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice,
eDice, eJaccard, Fager, Faith, Gower, Hamman, Jaccard,
Kulczynski1, Kulczynski2, Michael, Mountford, Mozley, Ochiai,
Pearson, Phi, Phi-squared, Russel, simple matching, Simpson,
Stiles, Tanimoto, Tschuprow, Yule, Yule2

* Distance measures:
Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean,
fJaccard, Geodesic, Hellinger, Kullback, Levenshtein, Mahalanobis,
Manhattan, Minkowski, Podani, Soergel, supremum, Wave, Whittaker
```

(Dis)similarity between groups

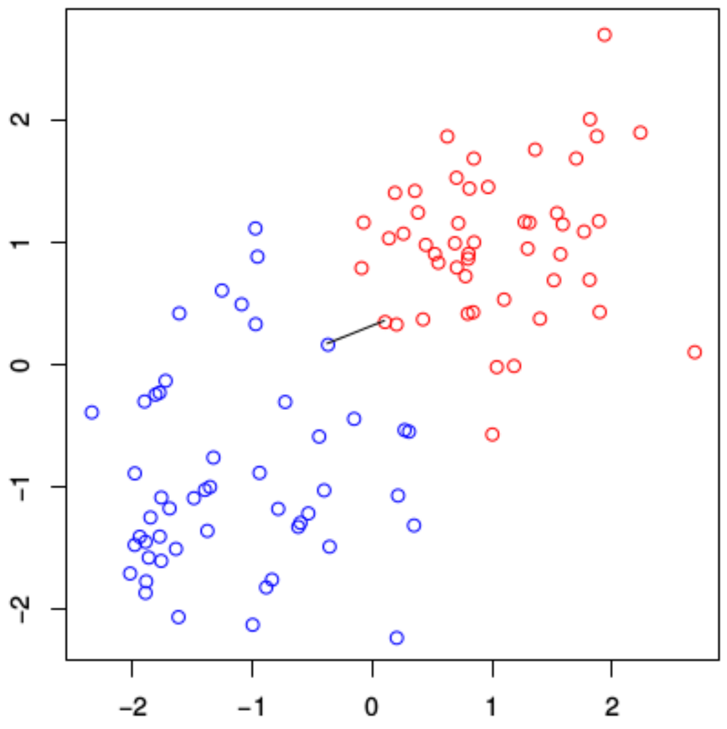
- **Single linkage** (nearest neighbour)
- **Complete linkage** (most remote neighbour)
- **Average linkage**

Single linkage

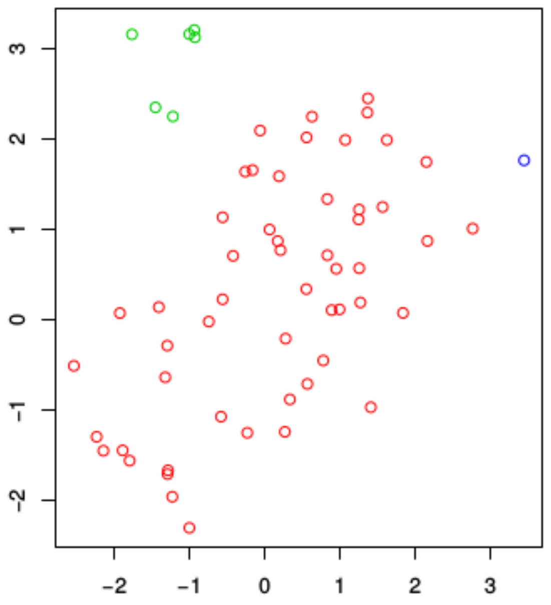
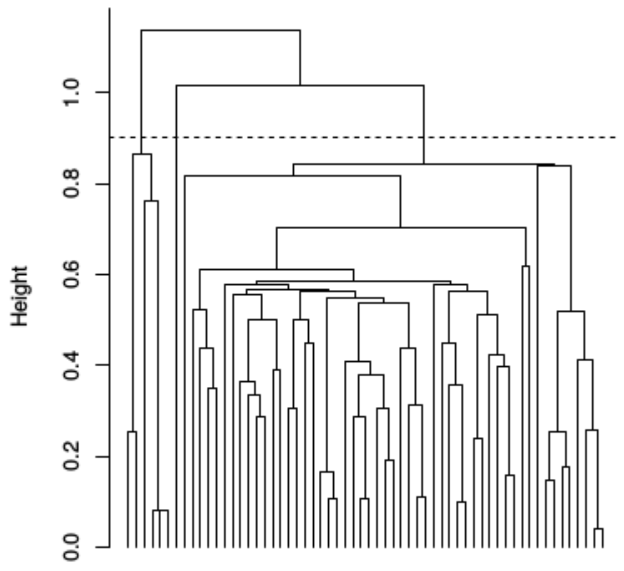
Single linkage (nearest neighbour): the *minimal* distance between the observations from different clusters:

$$d_{\text{Single linkage}}(C_k, C_{k'}) = \min_{i \in C_k, i' \in C_{k'}} d_{ii'}.$$

Concept



Example

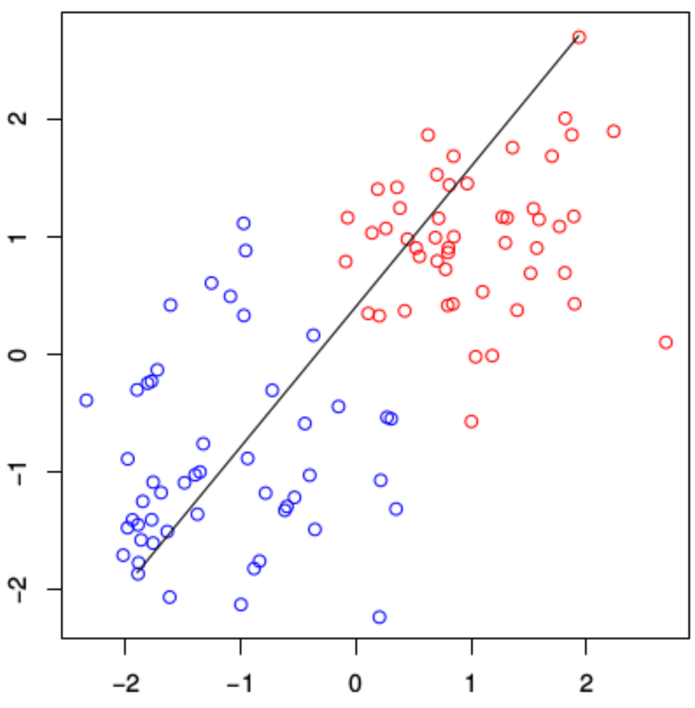



Complete linkage

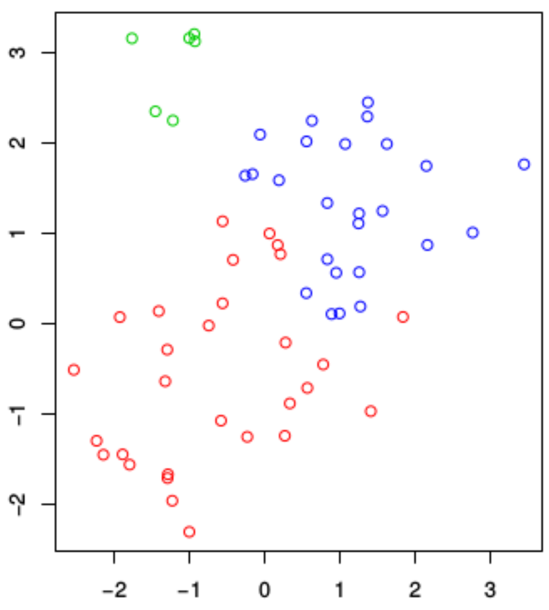
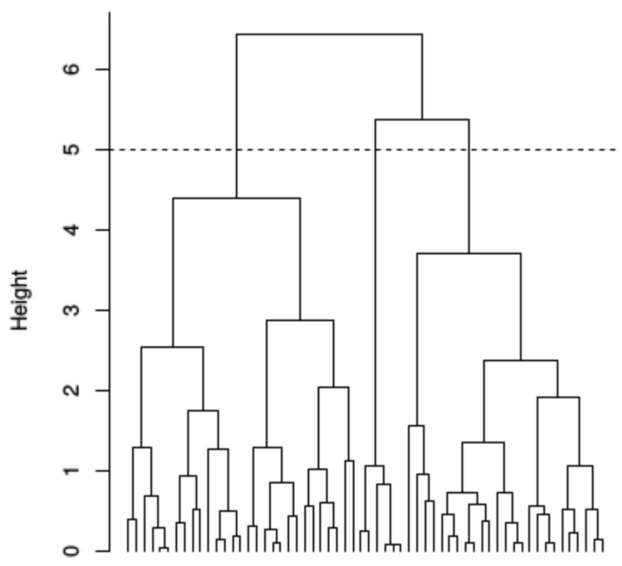
Complete linkage (most remote neighbour): the *maximal* distance between the observations from different clusters:

$$d_{\text{Complete linkage}}(C_k, C_{k'}) = \max_{i \in C_k, i' \in C_{k'}} d_{ii'}.$$

Concept



Example

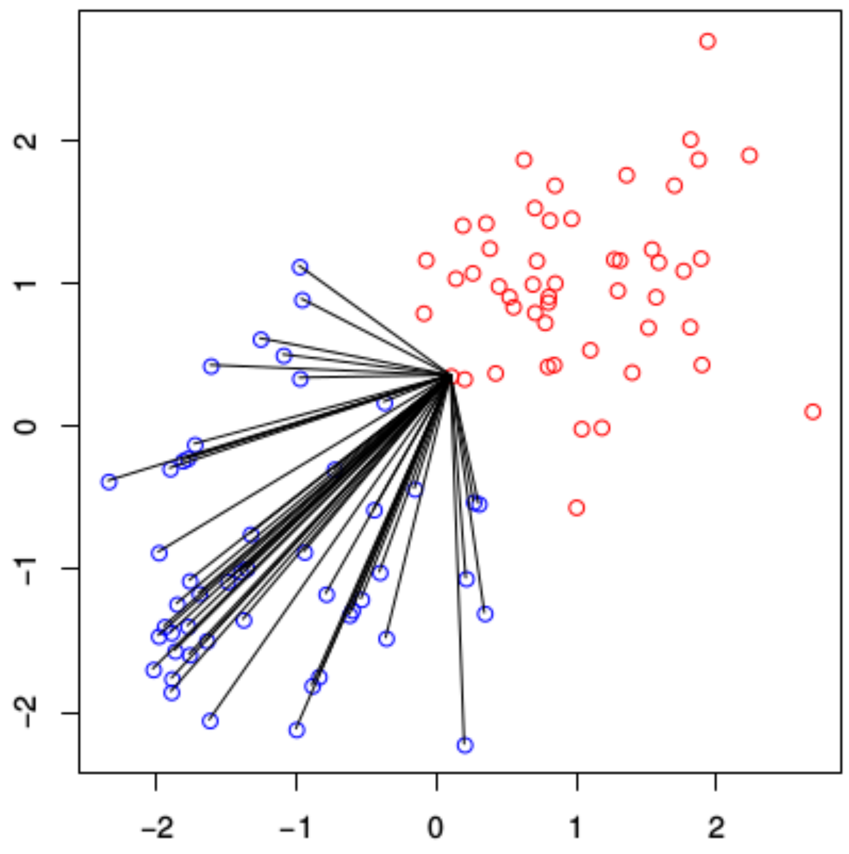



Average linkage

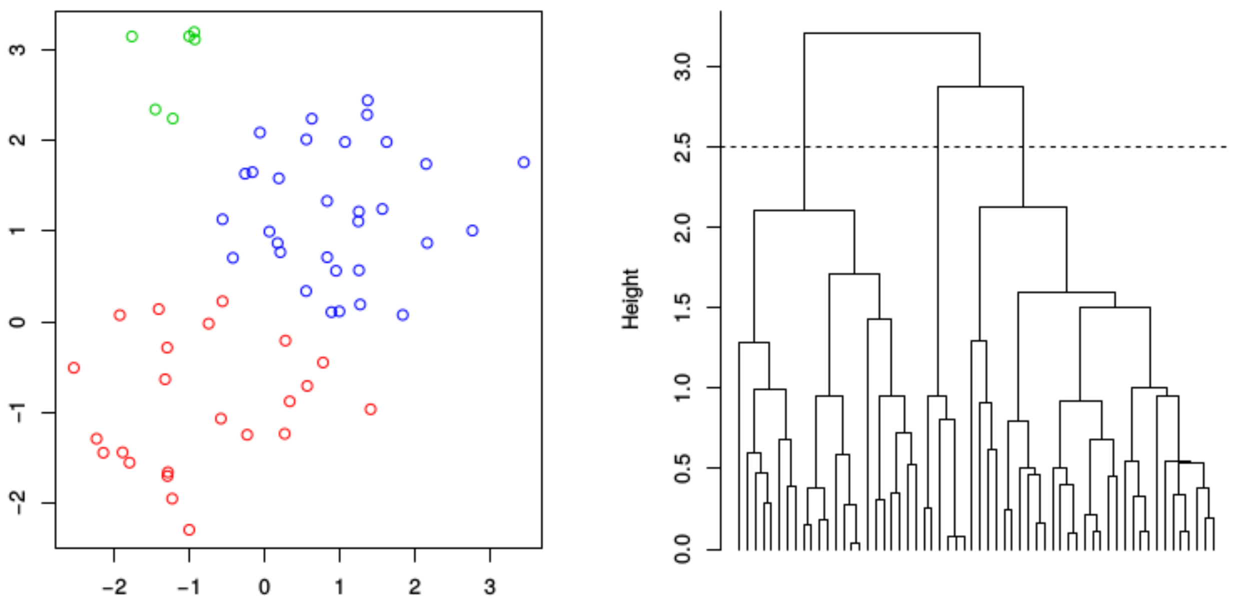
Average linkage: average distance between all pairs (one in each cluster)

$$d_{\text{Average linkage}}(C_k, C_{k'}) = \frac{1}{|C_k| \cdot |C_{k'}|} \sum_{i \in C_k, i' \in C_{k'}} d_{ii'}.$$

Concept



Example



Algorithm

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For $i = n, n-1, \ldots, 2$:
   - Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The

dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
    - Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters.

In R

- `hclust`
- `cutree`

Combine PCA and clustering

If we combine PCA with other techniques we often obtain better results. This is due to the sphering of the data done by PCA. E.g. for classification/regression trees the splits are always perpendicular to the axes (the variables in data). However, splitting may be more efficient if done using a sloped line through the data points. The PCA rotated data if therefore at times better to use – both for classification but also clustering

Example: Crabs

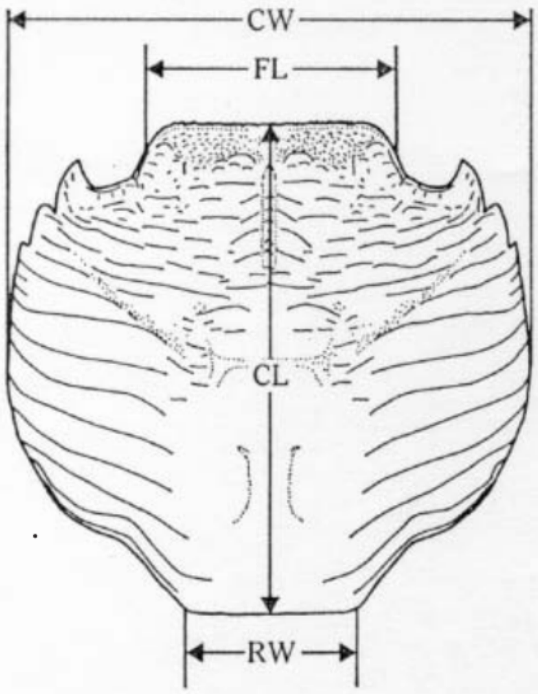
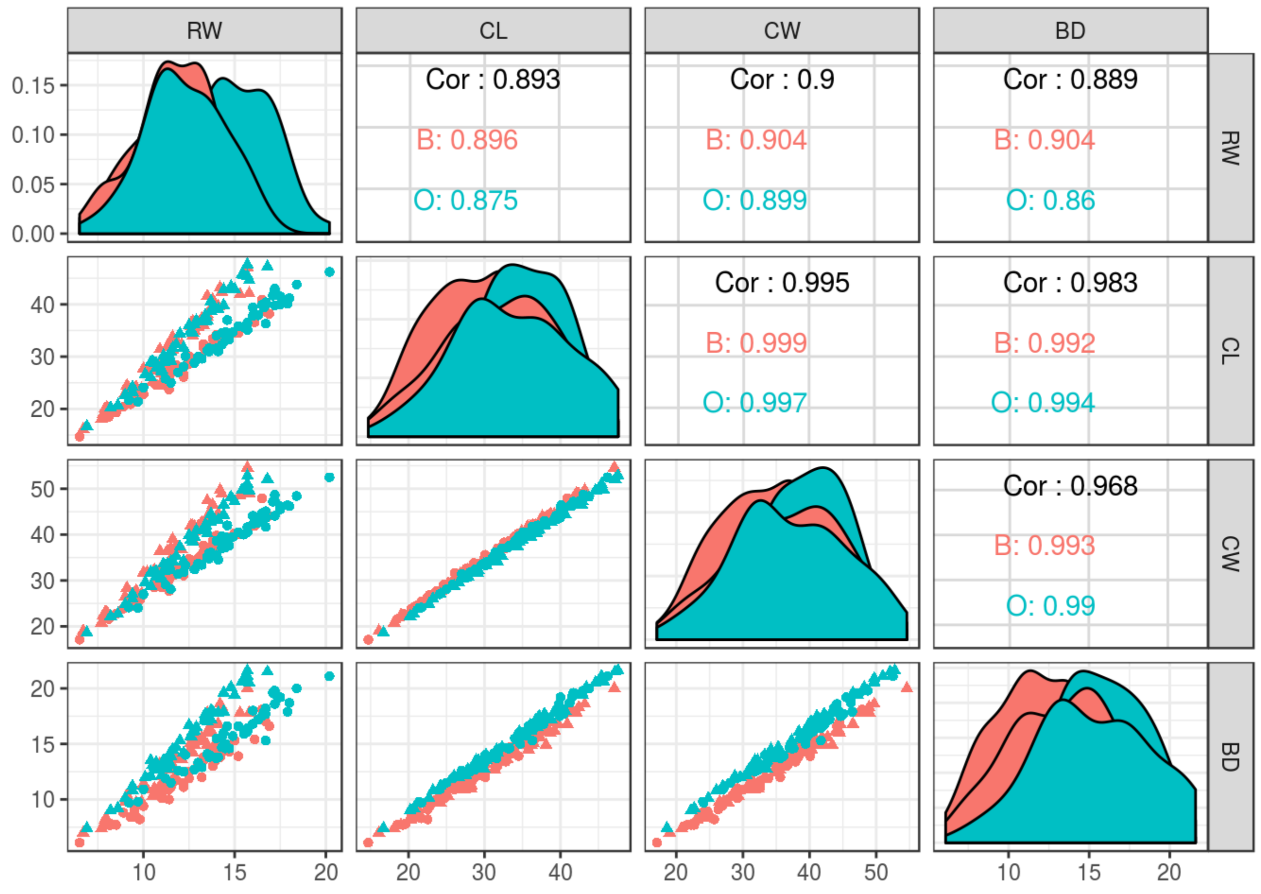


Fig. 1. Dorsal view of carapace of *Leptograpsus*, showing measurements taken. *FL*, width of frontal region just anterior to frontal tubercles. *RW*, width of posterior region. *CL*, length along midline. *CW*, maximum width. The body depth was also measured; in females but not in males the abdomen was first displaced.

Crabs

The dataset `data(crabs, package = "MASS")` contains five measurements done on 200 crabs of the species *Leptograpsus variegatus*. There are 50 female and 50 male crabs from each of two colours: blue and orange. Hence four groups of 50 crabs each.

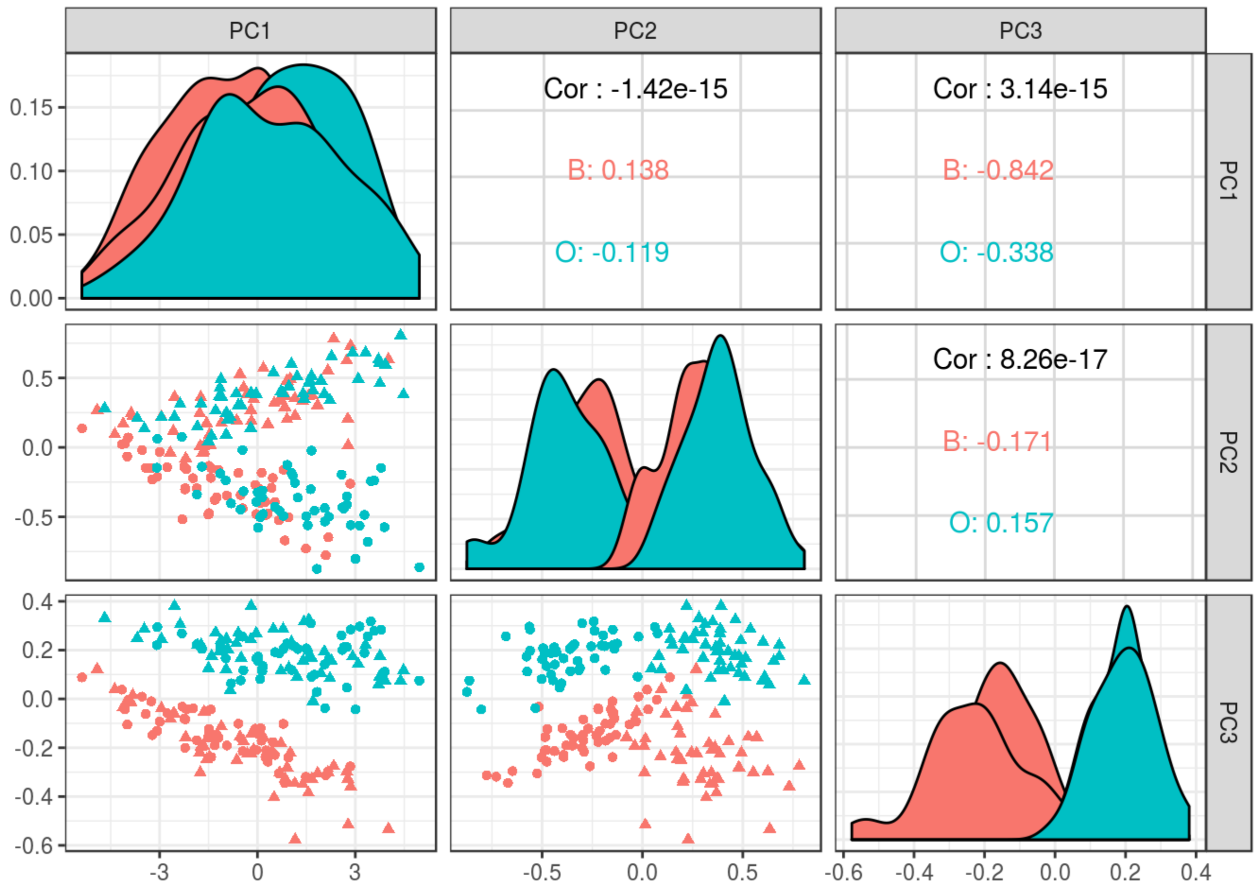


PCA

First we do a PCA on the numerical data

Then we join the PCA rotated data with the sample information

Then we make a plot of the first three PCs



K-means clustering

The K-means algorithm can then better identify the clusters in data

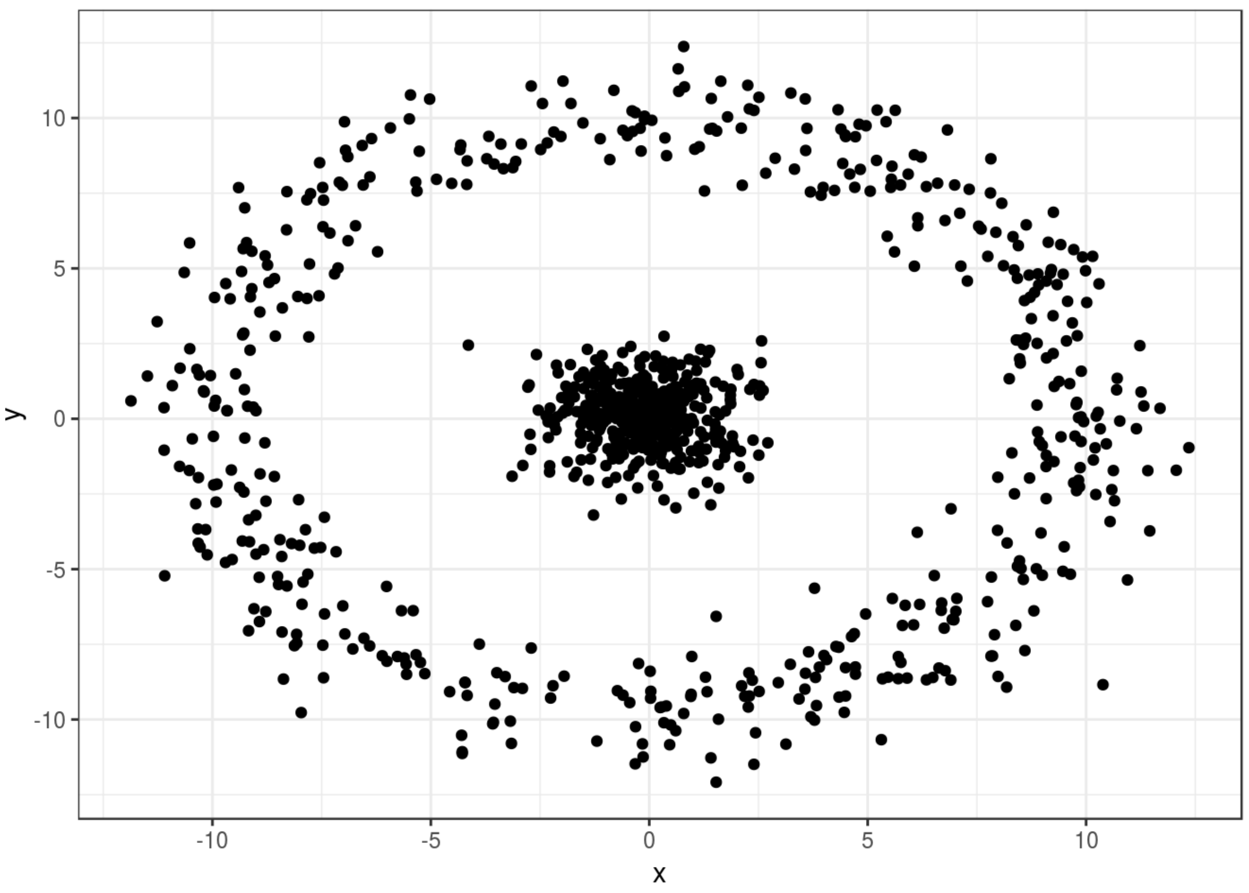
If we compare to the *true* structure in data

```
        sp_sex
cluster B_F B_M O_F O_M
      1  14  17  17  14
      2  17  10   4   6
      3  18  16  14  19
      4   1   7  15  11
```

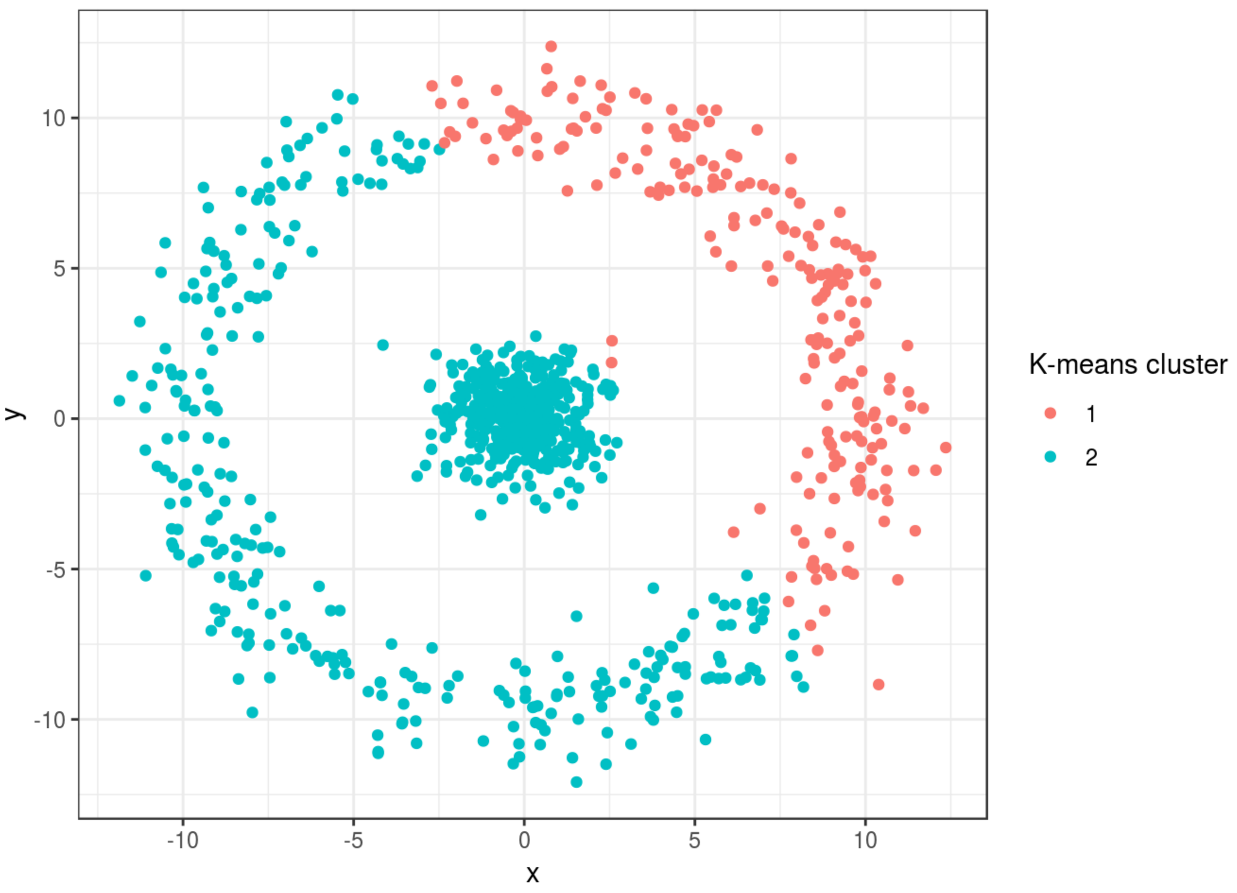
```
        sp_sex
cluster B_F B_M O_F O_M
      1   3   7   3  50
      2   1   0  46   0
      3  46   7   1   0
      4   0  36   0   0
```

Caveats

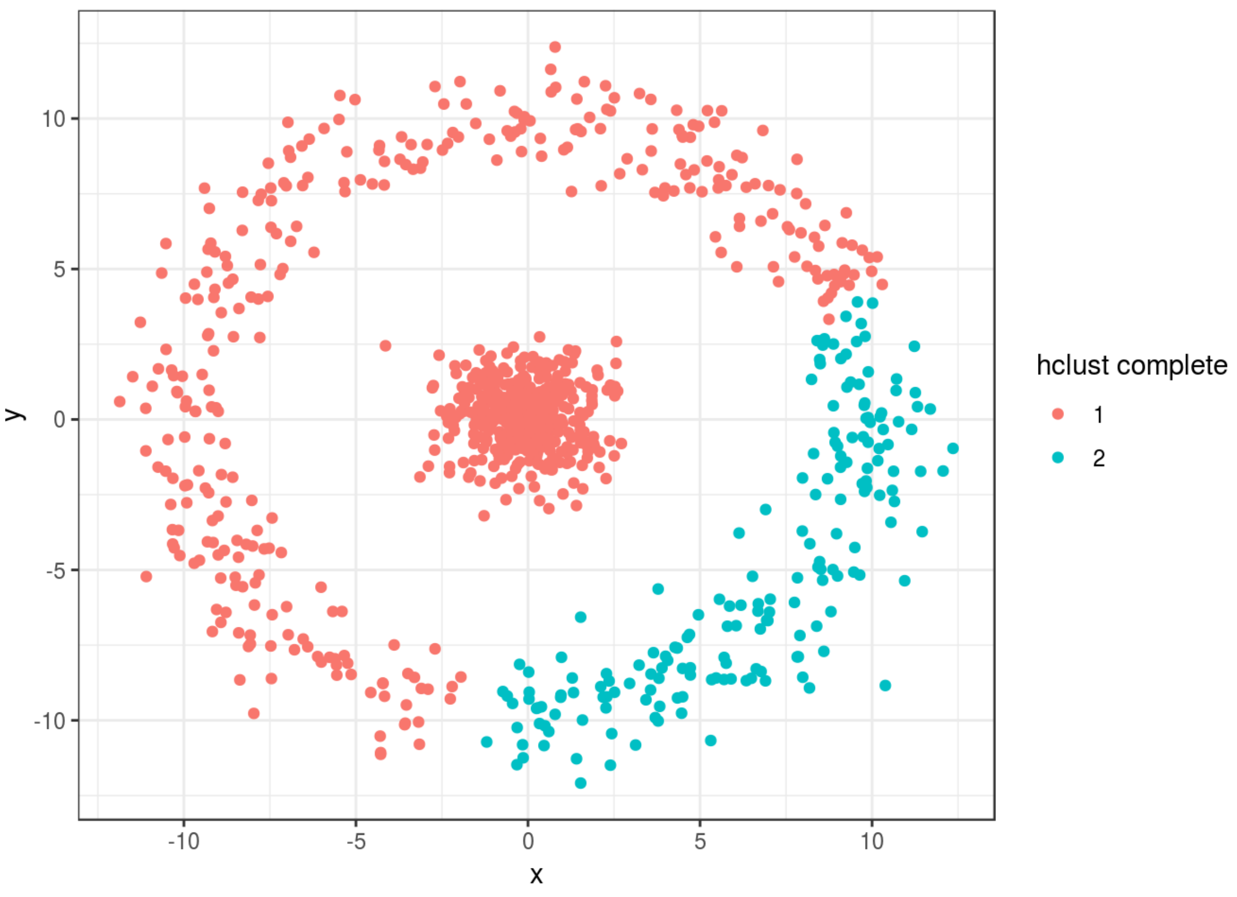
Caveat



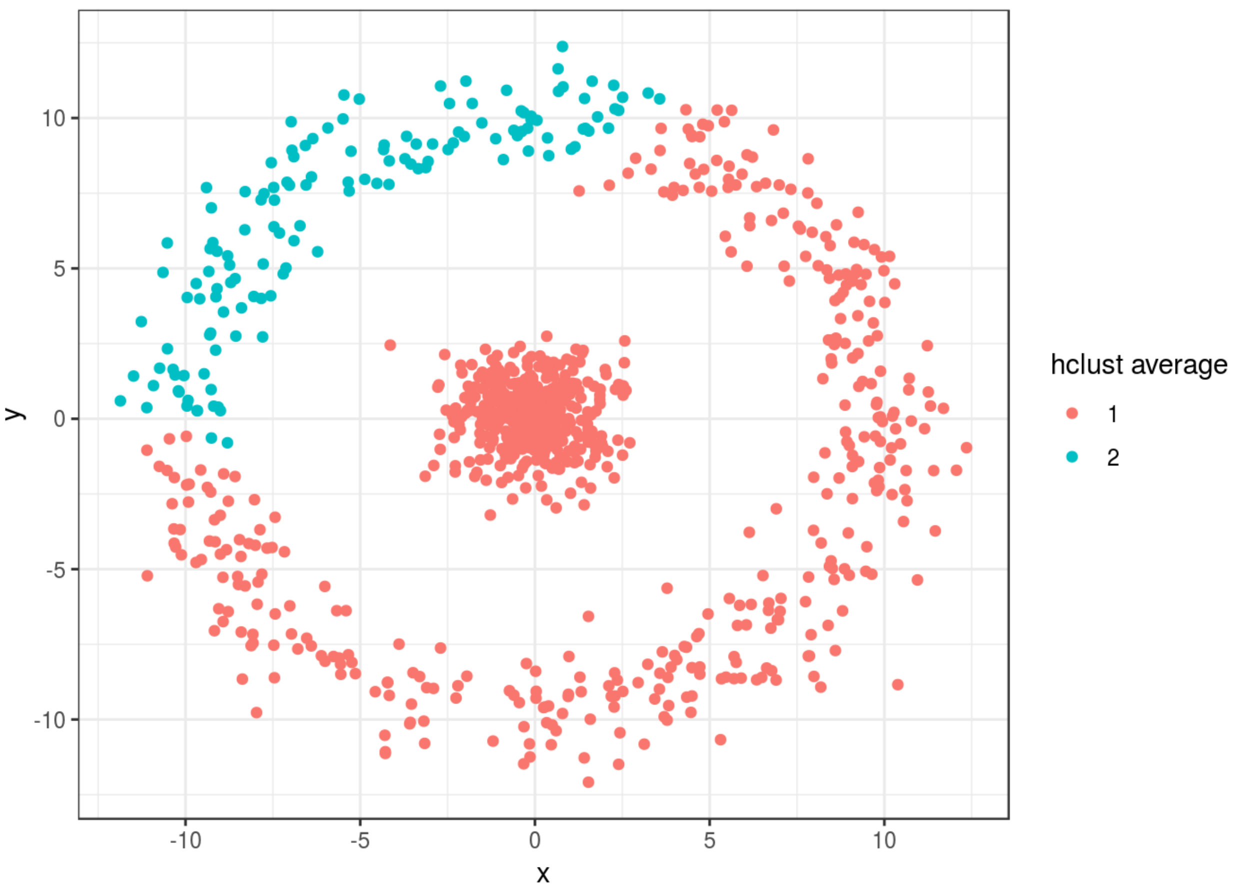
K-means



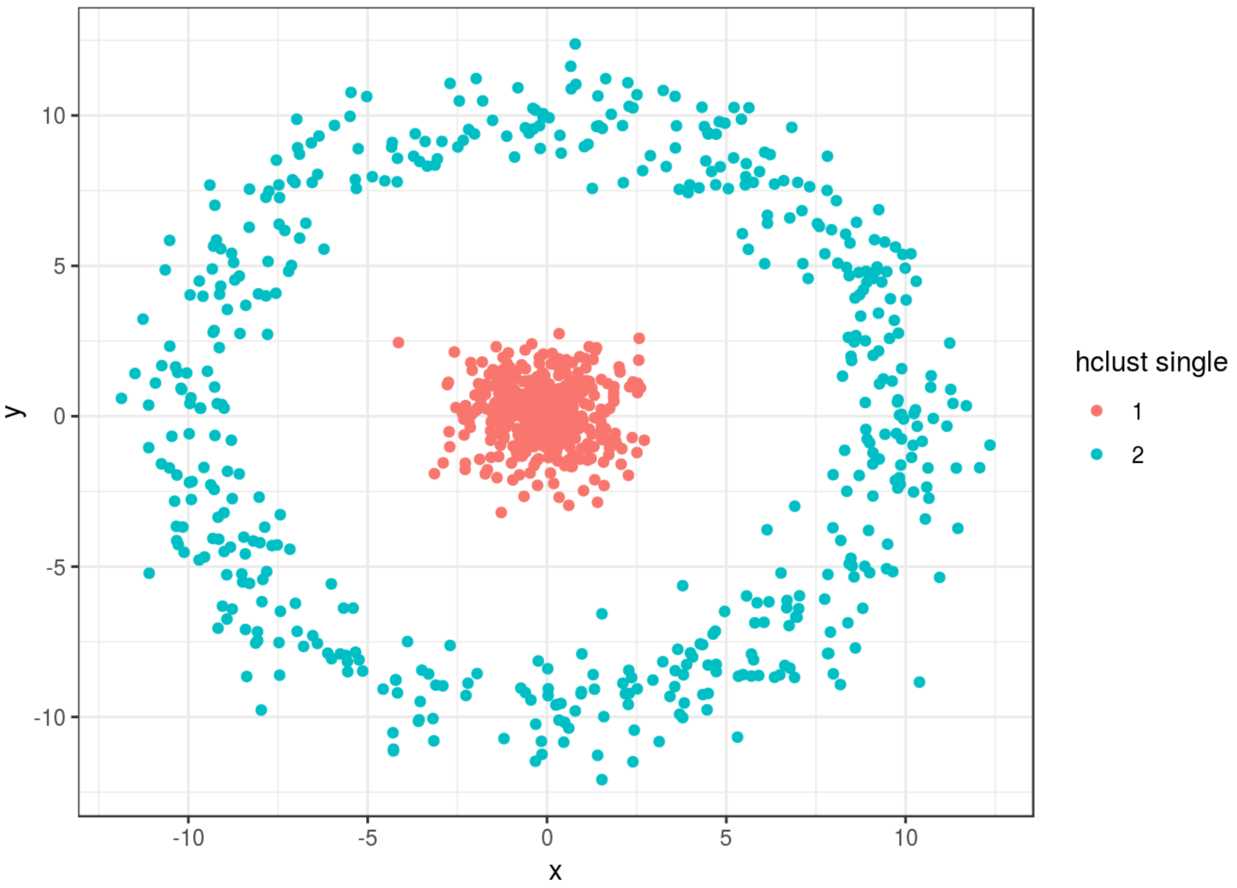
hclust with complete linkage



hclust with average linkage



hclust with single linkage



Caveats

K-means:

- Non-spherical clusters
- Unevenly sized clusters
- Clustering non-clustered data: uniform data will get divided into clusters
- SSE is worth minimizing / sensitive to scale
    - Different clusters can have different variance (`mclust`)

Generally, important choices with many options and large implications:

- (Dis)similarity between groups
- Distance between observations

Dendrograms:

Only the vertical distances are important - the horizontal ordering is (almost) arbitrary as there exists $2^n - 1$ orderings/permutations (chosen to be *pretty*).

Play around with dendrograms using `library(dendextend)`

Exercises

1. Do the example in `?kmeans`
    - Optional: Implement the Calinski and Harabasz index and find the best number of clusters
2. Do the example in `?cutree`
    - Do `plot(hc)` and use `rect.hclust`
3. Do 10.5.1 “K-means Clustering” (Lab 2) in ISLR (An Introduction to Statistical Learning, p. 404)
4. Do 10.5.2 “Hierarchical Clustering” (Lab 2) in ISLR (An Introduction to Statistical Learning, p. 406)
    - The last part on Correlation-based distance is optional