VICTORIA UNIVERSITY OF
WELLINGTON
TE HERENGA WAKA

AIML427

Clustering 2

Dr. Bach Nguyen

(with thanks to Prof. Bing Xue)

Bach.Nguyen@vuw.ac.nz

Outline of Week 6

- Supervised learning and Unsupervised learning
- Clustering analysis

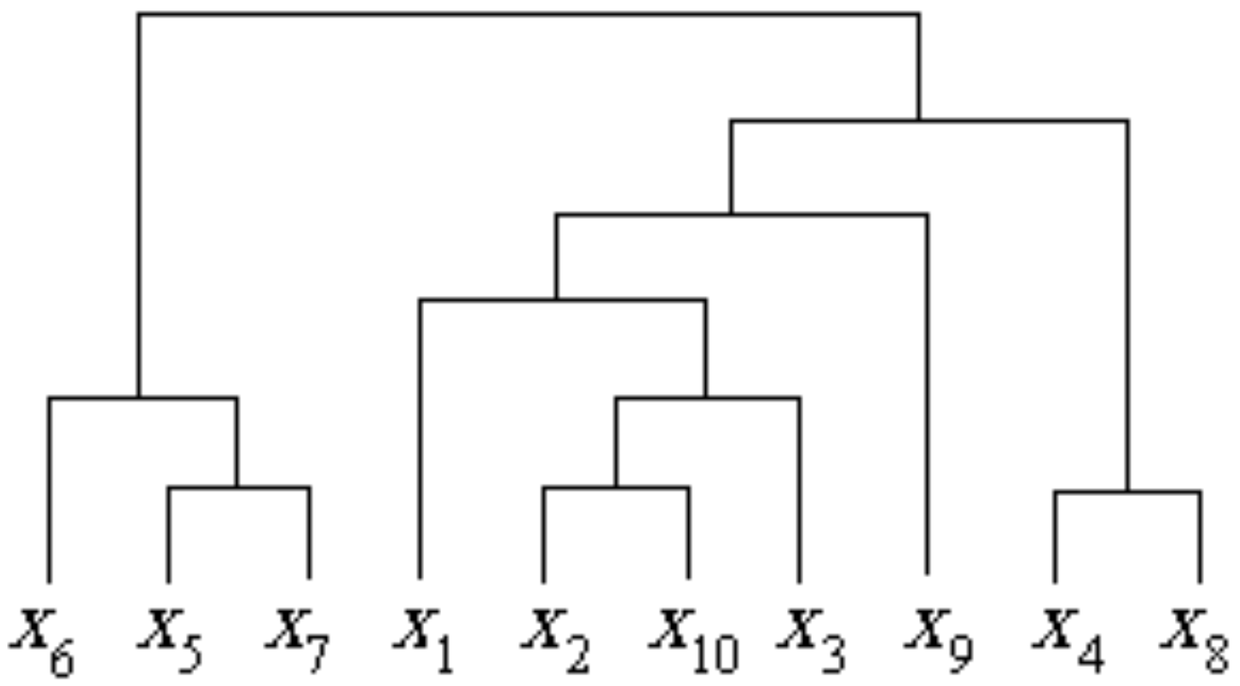
- Clustering Performance
- Clustering Metrics

To understand how to use and interpret:

- K-means clustering
- Hierarchical clustering
- Convex clustering

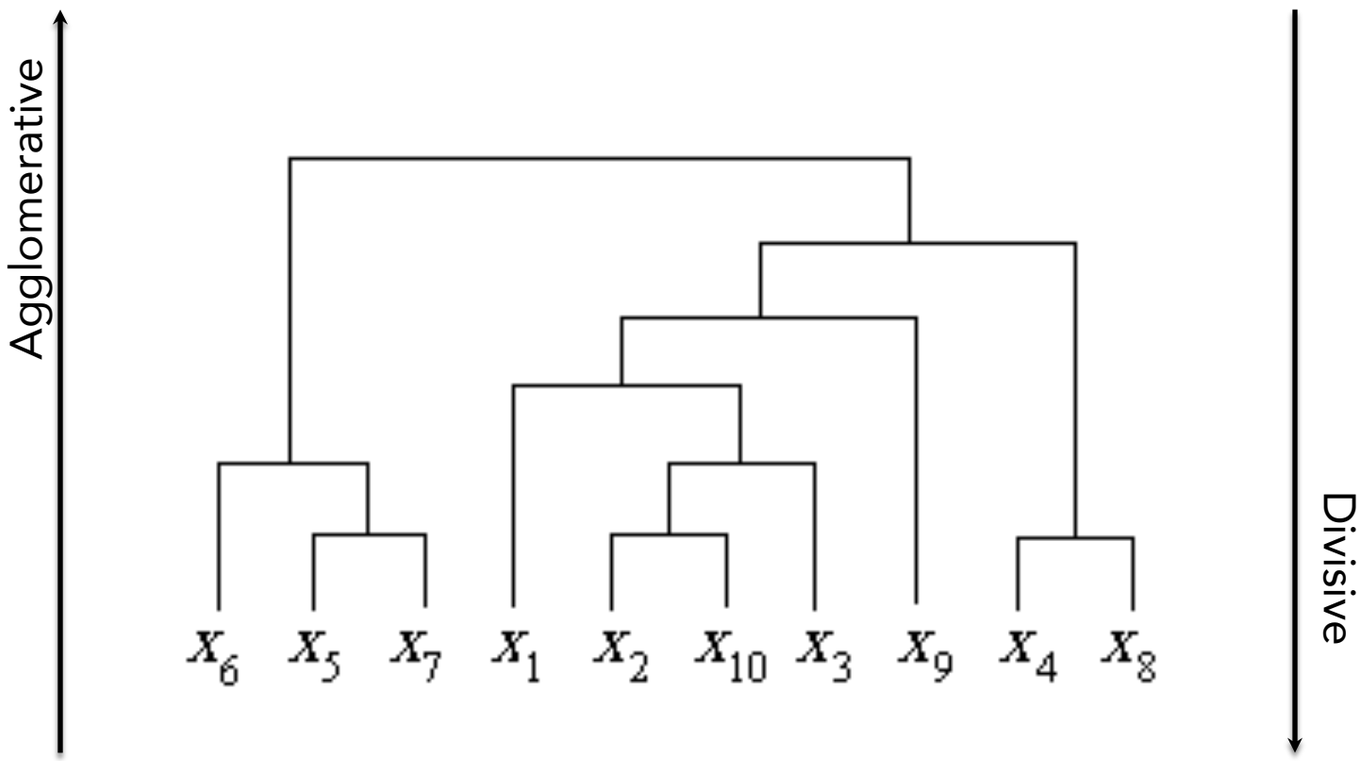
Hierarchical clustering

- The aim of **hierarchical clustering** is *a hierarchy of clusters*(!)
  - don’t commit to the number of clusters beforehand,
  - instead we obtain a *tree-based* representation of the observations known as a **dendrogram**
- See also ISLR 10.3.2



Hierarchical clustering

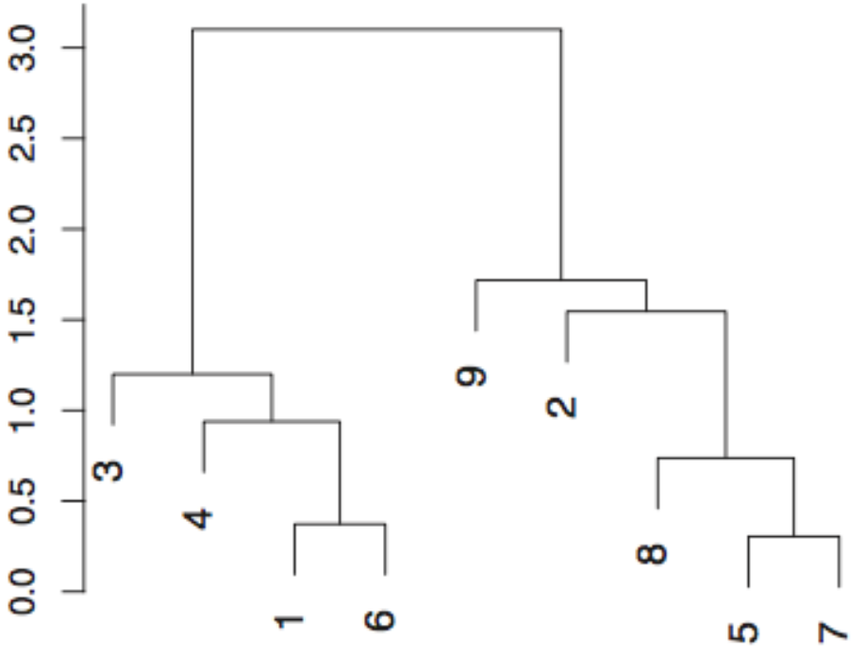
- There are two ways to do hierarchical clustering:
  - **Agglomerative** or **bottom-up** clustering where we start with the observations in *n* clusters – the leaves of the tree – and then merge clusters – forming branches – until there is only 1 cluster, the trunk of the tree
  - **Divisive** or **top-down** clustering where we start with the observations in 1 cluster and then split clusters until we reach the leaves
- We will focus on agglomerative clustering as it is generally much more efficient than divisive clustering

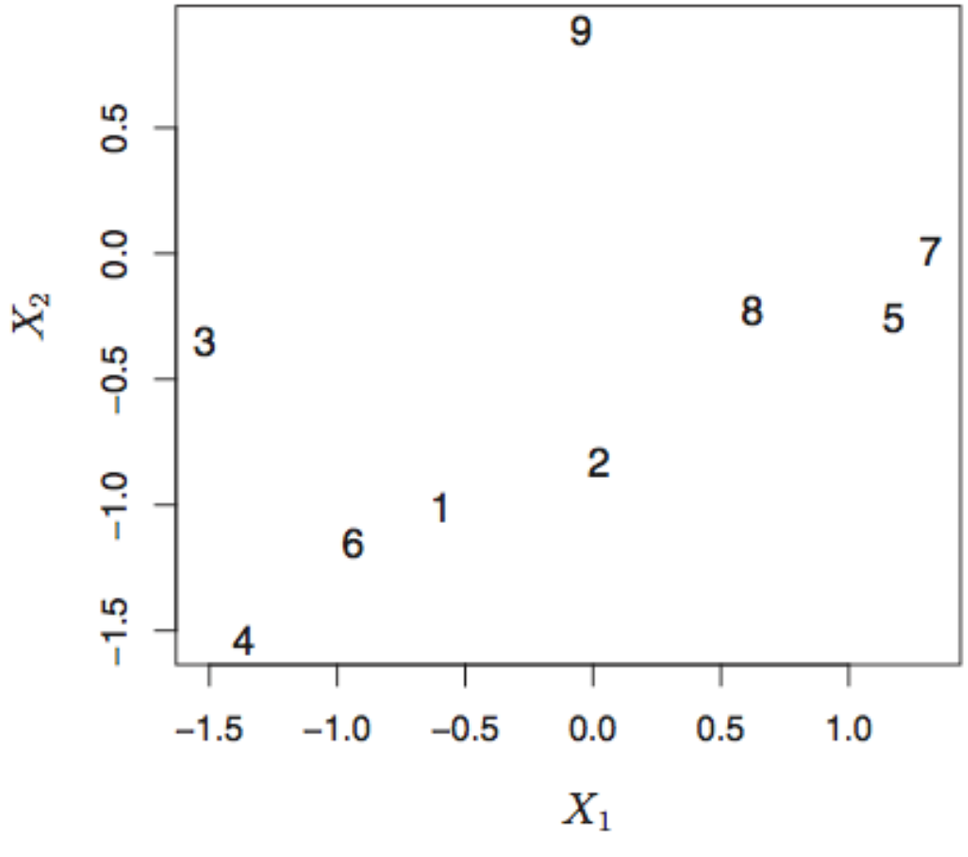


Interpreting a dendrogram

- Dendrograms are usually drawn with the leaves at the bottom and the trunk at the top

The dissimilarity between two observations is related to the vertical height at which they first get merged into the same cluster. The greater the height, the greater the dissimilarity

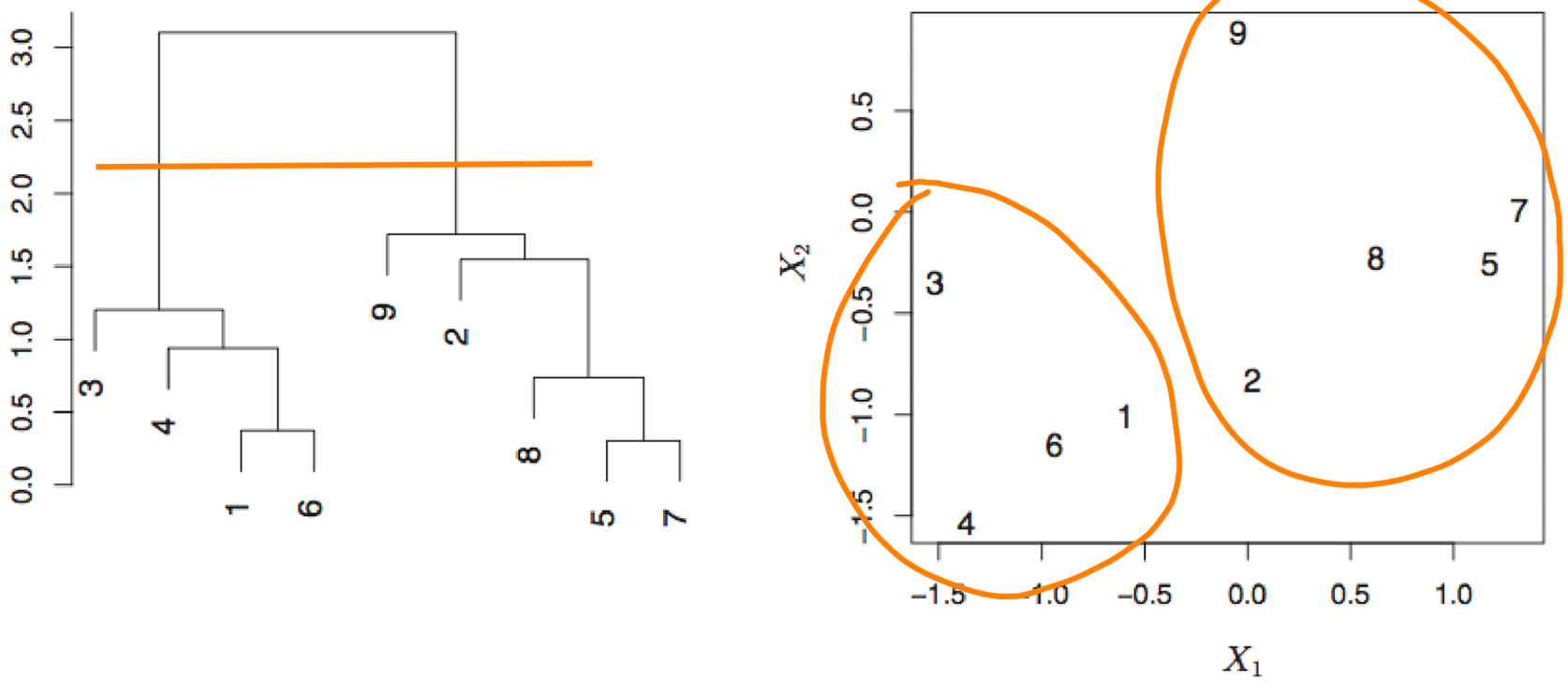




ISLR Figure 10.10: $n = 9$ and $p = 2$

Cutting a dendrogram

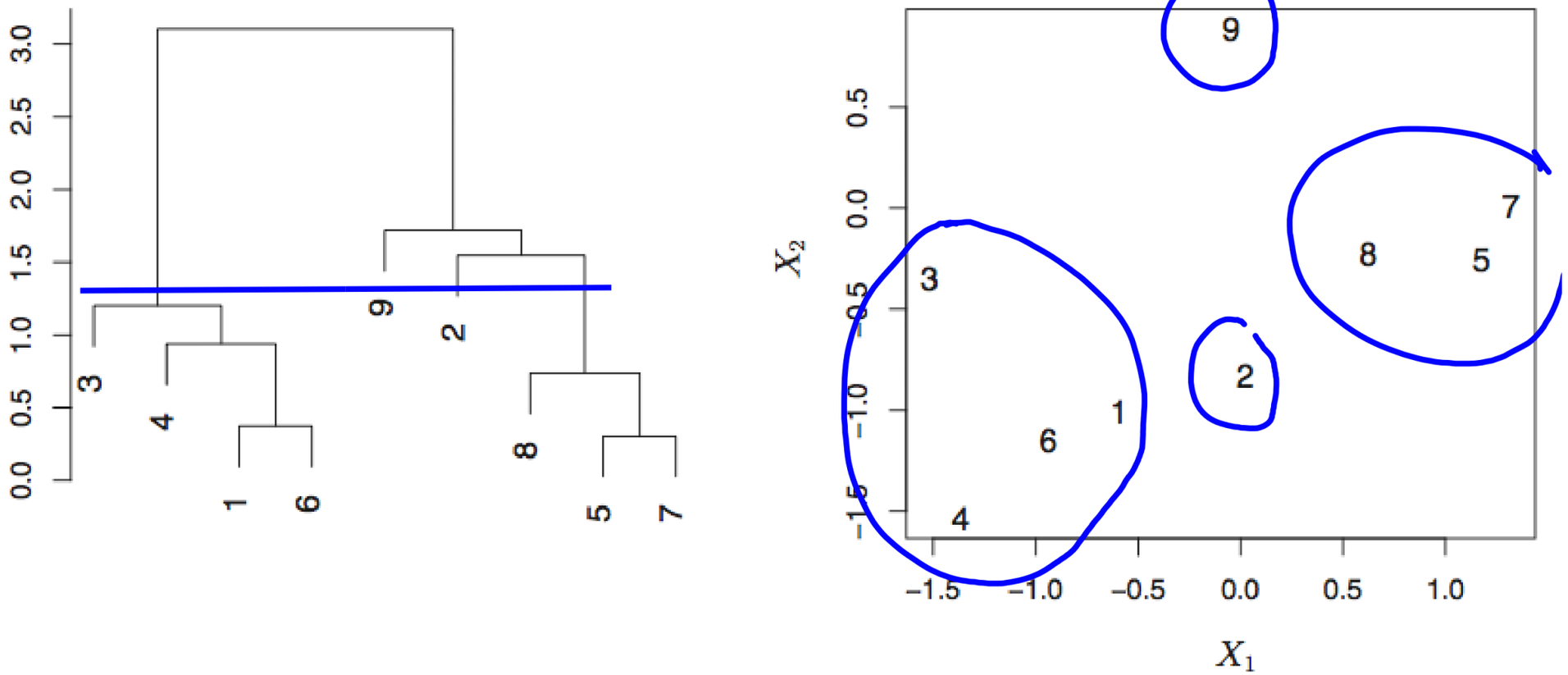
- Cutting a dendrogram horizontally gives a natural clustering. The height of the cut determines the number of clusters



2 clusters

Cutting a dendrogram

- 4 clusters



4 clusters

Dissimilarity measure and linkage

- The two key ingredients/components of hierarchical clustering are a dissimilarity measure and a linkage method:
  - The dissimilarity measure quantifies *how dissimilar a pair of* ***observations*** *are.*
  - Linkage tells us *how to extend the dissimilarity measure to pairs of* ***clusters****.*
  - The *choices* of dissimilarity measure and **linkage method** have *profound effects* on the resulting clustering

Dissimilarity measures

Here are some common dissimilarity measures *d(x,y)* for observations x,y:

- Euclidean distance: $\sqrt{(x-y)^2}$
- Squared Euclidean distance: $(x-y)^2$
- Manhattan distance: $\sum_{j=1}^{p} |x_j - y_j|$
- Maximum distance: $\max_j |x_j - y_j|$
- Correlation-based distance: $1 - corr(x, y)$

For text or non-numeric data other distances can be defined, e.g. the Hamming distance

Linkage methods

The linkage method quantifies the dissimilarity between clusters A and B.

- *how to represent each cluster* ? -- to tell which pair of clusters is closest

The standard linkage methods include:

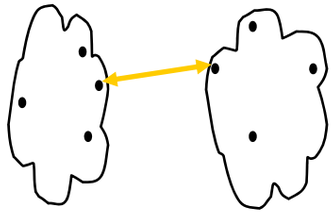
- **Centroid**: dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B.
- **Complete**: compute the maximum pairwise dissimilarity where one observation is in cluster A and the other is in cluster B
- **Single**: compute the minimum pairwise dissimilarity where one observation is in cluster A and the other is in cluster B.
- **Average**: compute the average pairwise dissimilarity where one observation is in cluster A and the other is in cluster B

Linkage methods

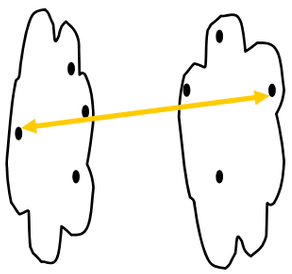
Generally,

- Centroid linkage can result in undesirable inversions.
- Complete and average linkage produce more balanced dendrograms;
- Single linkage can produce trailing clusters in which single observations are merged one-at-a-time

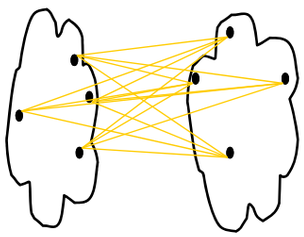
Single Linkage



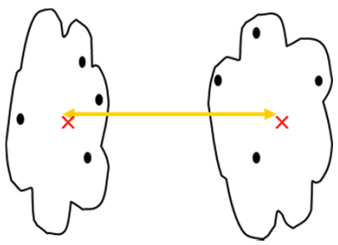
Complete Linkage



Average Linkage



Centroid



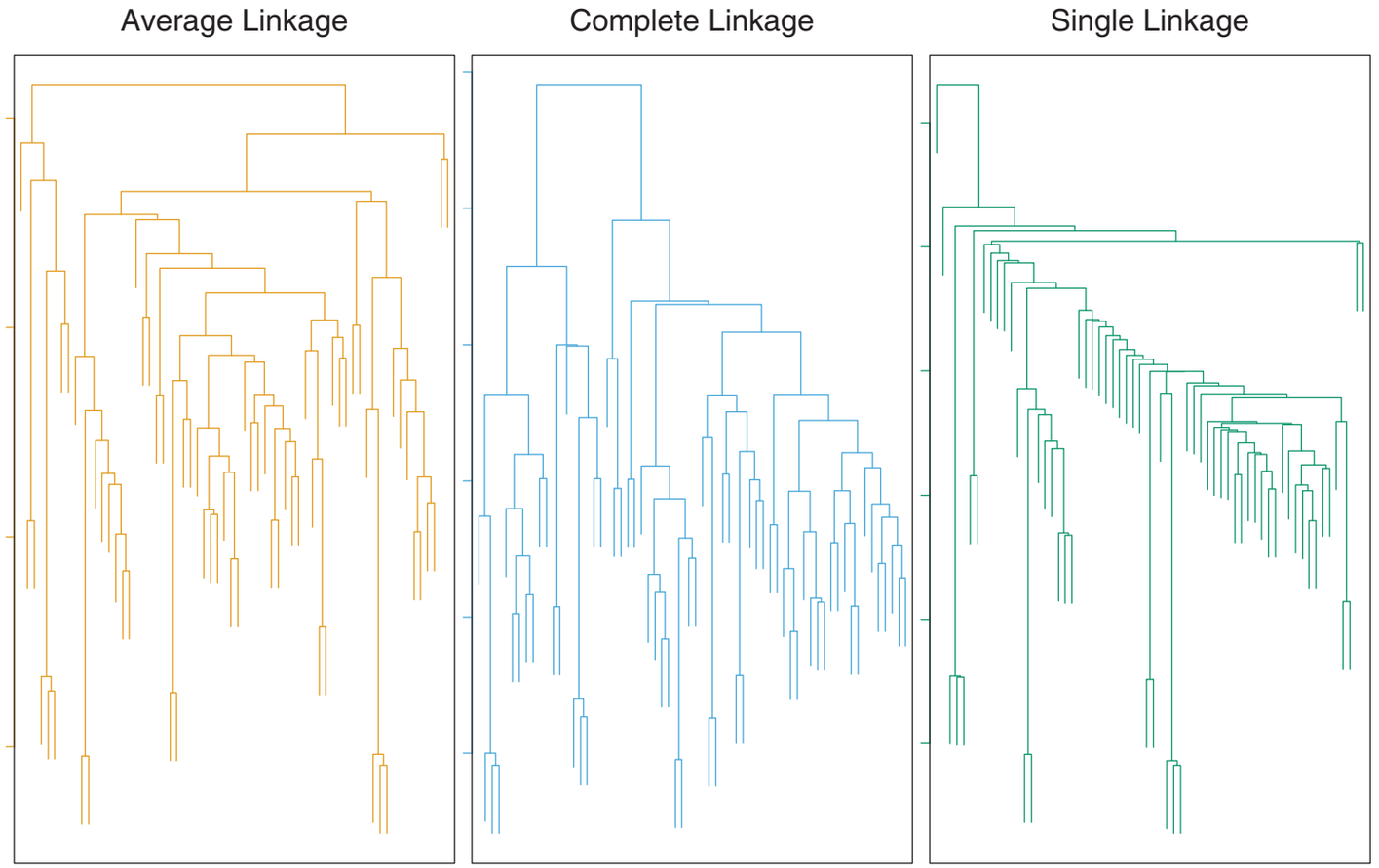


FIGURE 10.12. *Average, complete, and single linkage applied to an example data set. Average and complete linkage tend to yield more balanced clusters.*

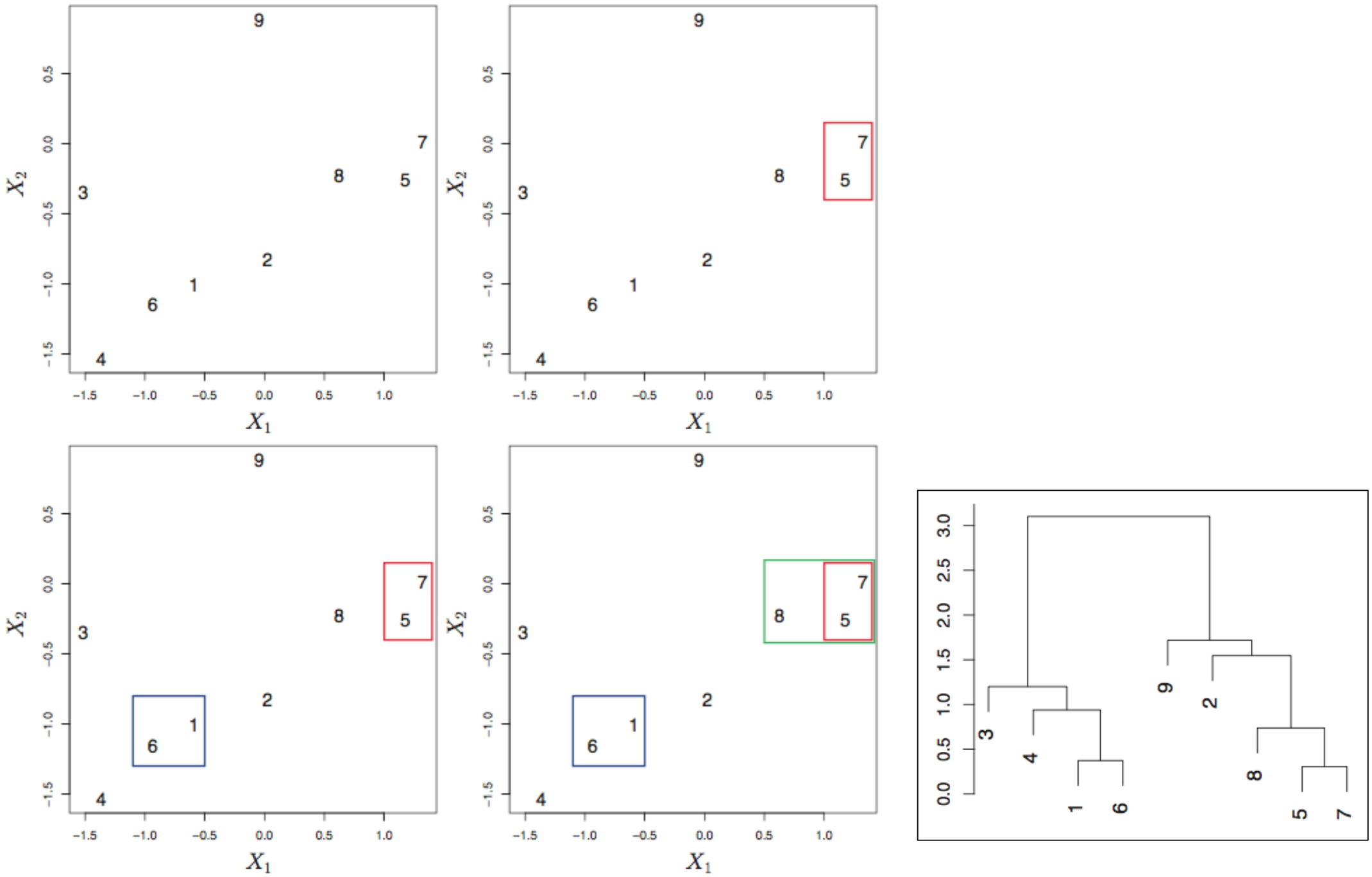
Agglomerative clustering algorithm

With the choice of dissimilarity measure and linkage method, **agglomerative clustering proceeds** as follows:

- Treat each observation as its own cluster, n clusters. Compute all pairwise dissimilarities (such as Euclidean distance) of all the $\binom{n}{2} = \frac{n(n-1)}{2}$ pairwise dissimilarities.
- For $i = n, n-1, \ldots 2$
  - (a) Find the pair of clusters that are the least dissimilar and merge them
    - The dissimilarity between these two clusters indicates the height on the dendrogram where the merge is shown.
  - (b) Compute all pairwise dissimilarities between the *(i-1)* remaining clusters

Note that there is no random initialisation, so agglomerative clustering is a deterministic algorithm

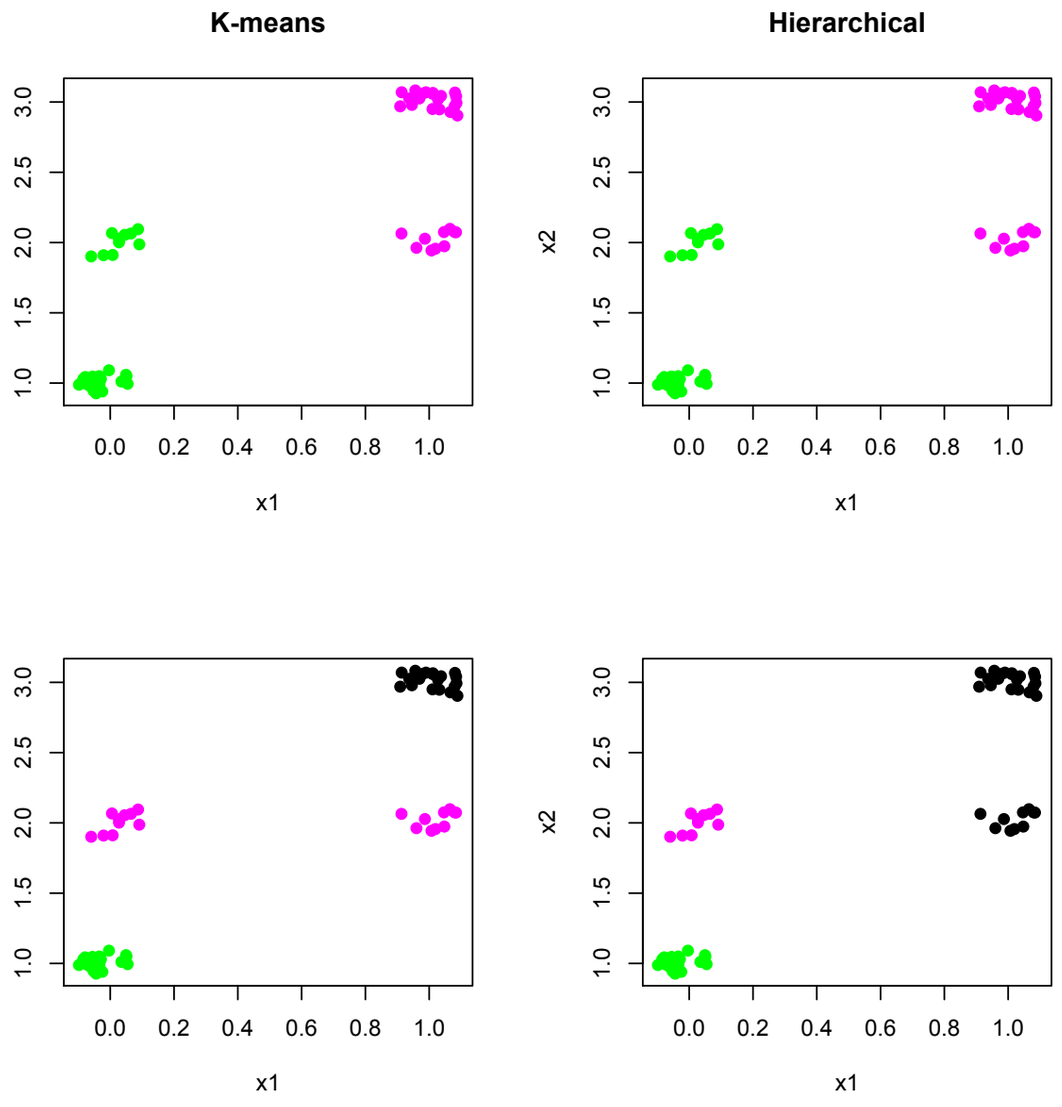
Agglomerative clustering algorithm



ISLR Figure 10.11: First few steps of the agglomerative clustering algorithm

A drawback of hierarchical clustering

- A *potential drawback* of hierarchical clustering is that clustering obtained by cutting the dendrogram at a certain height is necessarily ***nested*** within the clustering obtained by cutting at a greater height

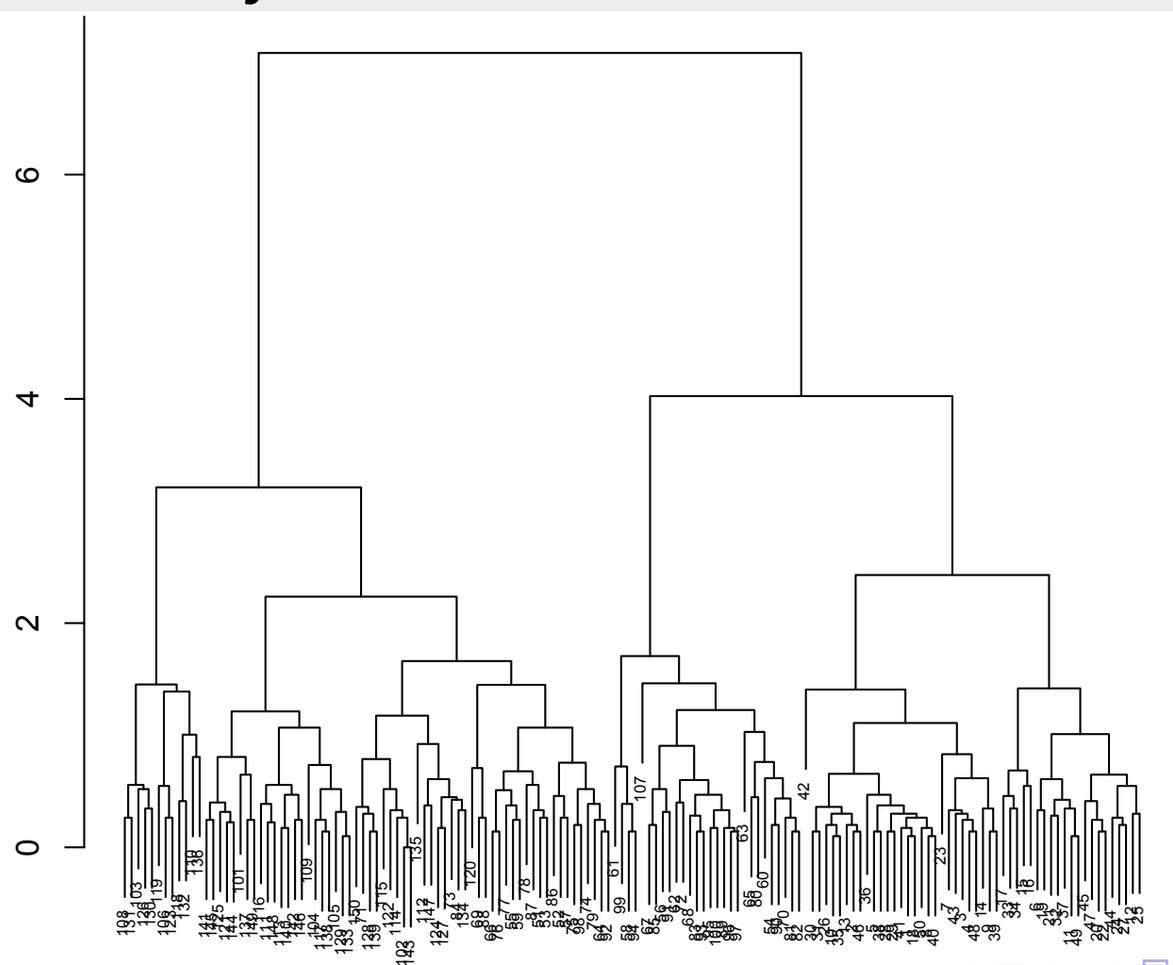


- Example: n = 60, p = 2; best 3-clustering /= split of best 2-clustering

Iris example

- The function *hclust* performs hierarchical clustering in R.
- However, it requires the matrix of all pairwise dissimilarities. To find this, we use the function *dist*.
- By default, *dist* uses Euclidean distance and *hclust* uses complete linkage

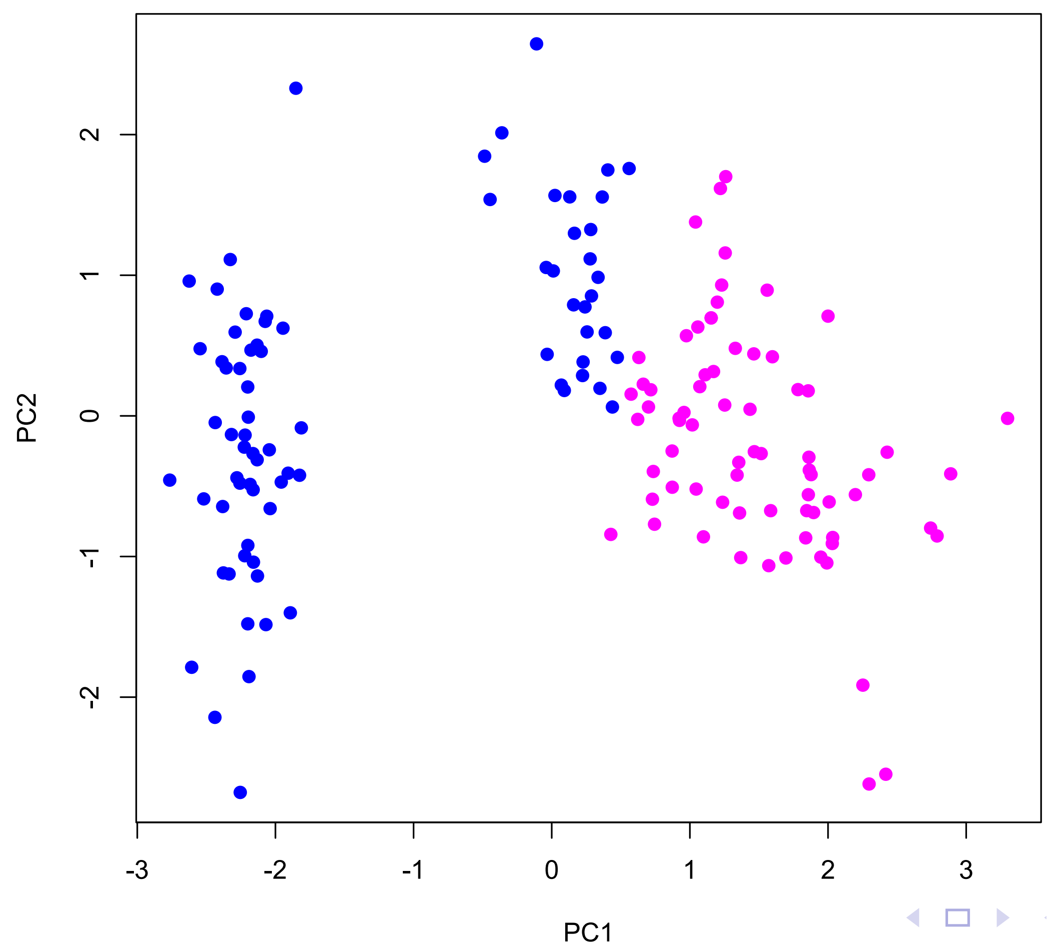
```
> hc = hclust(dist(iris[,1:4]) >
plot(hc,main="",xlab="",ylab="",sub="",cex=0.5)
```



Iris example

To obtain a clustering with a specified number of clusters, we use *cutree*. For $K = 2$:

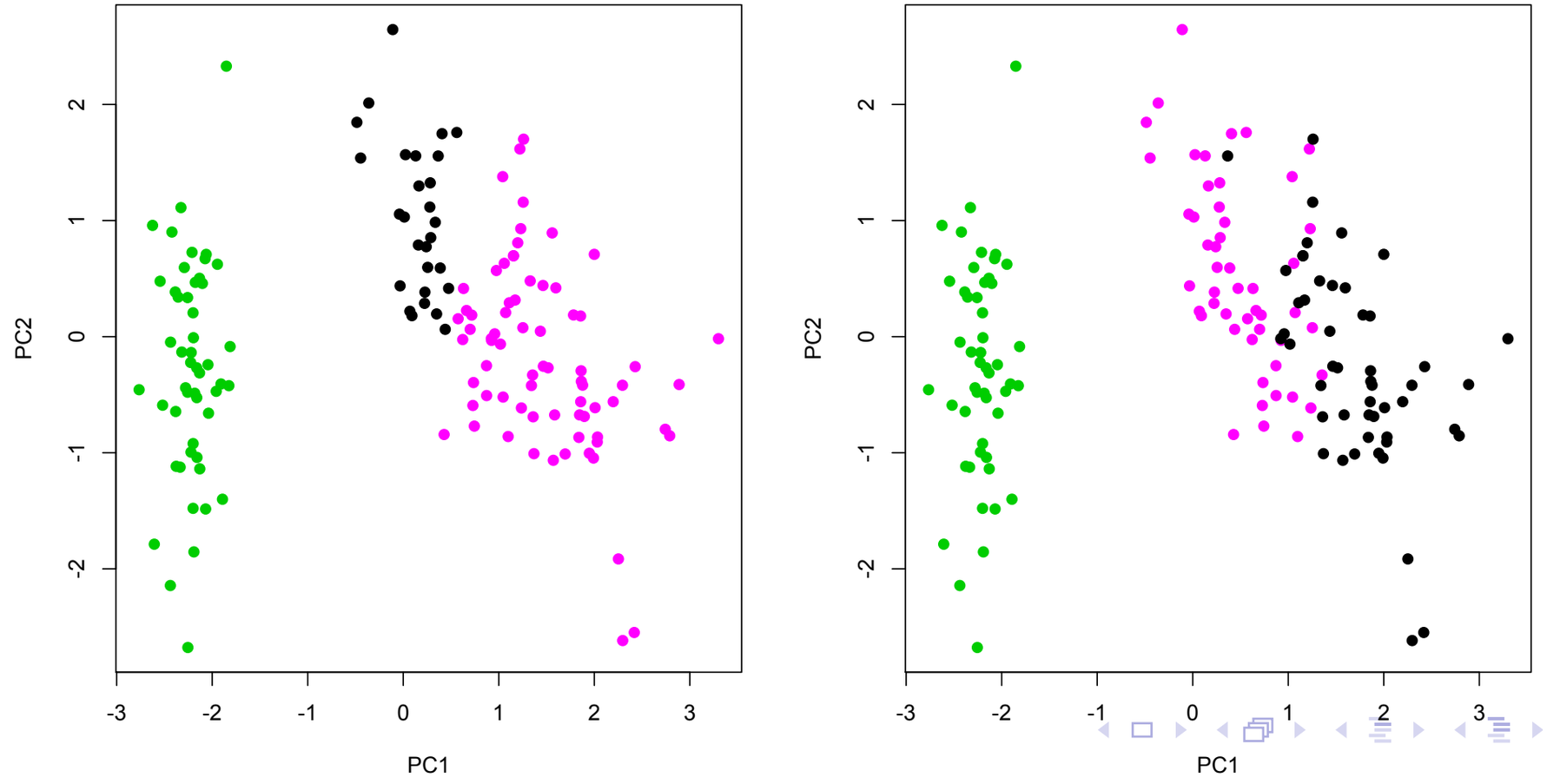
```
> cutree(hc, 2)

[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 1 1 1 1 1 1 1
[41] 1 1 1 1 1 1 1 1 1 1 2 2 2 1 2 1 2 1 2 1 1 1 1 2 1 2 1 1 2 1 2 1 2 2 2 2 2 2 2 1
[81] 1 1 1 2 1 2 2 2 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[121] 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 2 2 2 2 2
```



Iris example

When can compare the $K = 3$ clustering with the actual labels:

```
> table(cutree(hc,3), iris[,5])
setosa versicolor virginica
1     50          0         0
2     0          23        49
3     0           27        1
```



Comments on clustering

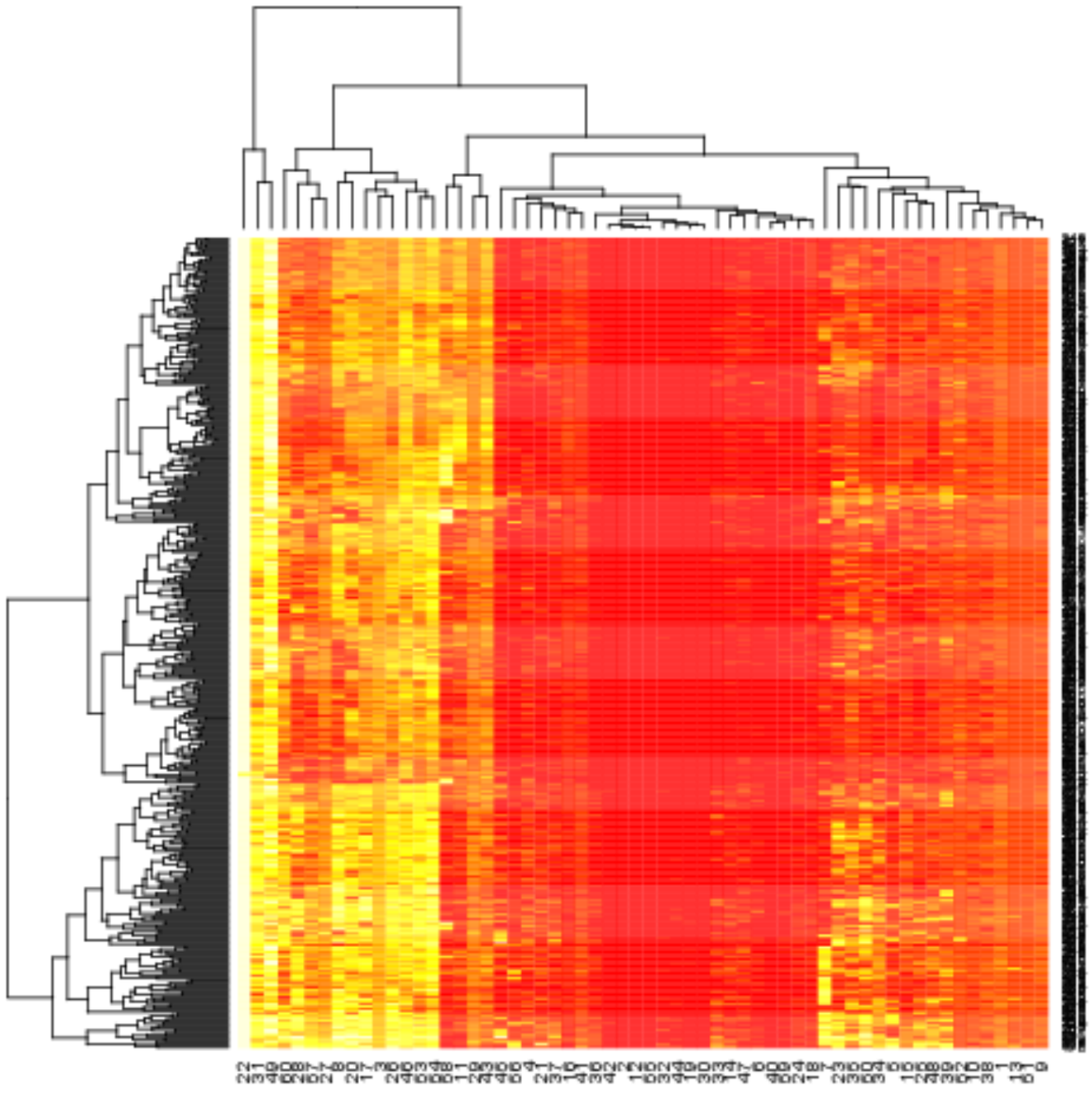
- Specifically, for hierarchical clustering:
  - dist(X,method="...") allows you to specify the dissimilarity measure in R
  - Similarly, hclust(d,method="...") allows you to specify the linkage method
  - As with K-means, consider standardising X

- More generally,
  - For a given dataset, *try various clustering approaches* to see which patterns consistently turn up
  - Check for robustness by clustering on subsets of the data; note however that clustering can *be quite sensitive* to small changes in the data
  - Use clustering as a tool to motivate further enquiry;
  - don’t overstate conclusions!

Heatmap

- Finally, R has a nice way of displaying a data matrix that has been hierarchically clustered

```
> heatmap(X, scale="none")
```



Heatmap

- Note that, by default, *heatmap* does hierarchical clustering on columns and rows; it then moves the columns and rows around to "cluster" the matrix
- *heatmap* has a large number of options that are worth exploring
- *heatmap.2* in the gplots library has even more!
- A *heatmap* can be unwieldy if *n* or *p* become large

Convex clustering

- Convex clustering is a *penalised* method for clustering that *has elements of K-means and hierarchical clustering*. The objective function is to minimise:

$$\sum_{i=1}^{n}(x_i - u_i)^2 + \lambda \sum_{i<j}^{n} w_{ij}(u_i - u_j)^2$$

where u_i is the clustering centroid for observation i, λ is a penalty parameter and w_{ij} are non-negative weights.

- If $\lambda = 0$, $u_i = x_i$, so that every observation is its own cluster
- If $\lambda \to \infty$, $u_i = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ for all i, so there is just one big cluster

Convex clustering

$$\sum_{i=1}^{n}(x_i - u_i)^2 + \lambda \sum_{i<j}^{n} w_{ij}(u_i - u_j)^2$$

- As λ increases, centroids "fuse".
  - The associated observations are now thought of as being in the same cluster, i.e. we have a type of agglomerative clustering
  - Obtain a tree-like clustering as λ varies
- Since this is *unsupervised* learning, there is no way to specify the "correct" value of λ
- The ℓ_2-*norm* can be replaced by the ℓ_1-*norm* in the penalty term (though the algorithm runs more slowly)
  - Ridge regression and Lasso regression

Choosing the weights in convex clustering

The weights w_{ij} are very important to the quality of the clustering and the speed of the algorithm.

- Naively, we might want to choose $w_{ij} = 1$ but this means clustering in widely separated regions will be linked.

The maintainers of the *cvxclustr* package in R recommend the following:

$$w_{ij} = \begin{cases} \exp\left(-\emptyset\left(x_i - x_j\right)^2\right), & i \text{ is one of the } K - \text{nearest neighbours of } j, \text{or vice versa} \\ 0, & otherwise \end{cases}$$

- Increasing $\emptyset$ or decreasing K makes the clustering more sensitive to the local density of data
- The dependence of the clustering on these additional parameters is simultaneously a frustration and an *opportunity*!

Mammal teeth example

- 8 kinds of teeth (features) ; 27 mammals (observations). Since we are dealing with the same kind of features, we will center but not scale them.

```
> library(cvxclustr)
> data(mammals)
> head(mammals)

           X topincisors bottomincisors topcanines bottomcanines toppremolars
1   oppossum           4              5          2             2            4
2  htailmole           4              4          2             2            5
3 commonmole           4              3          2             1            4
5   brownbat           3              4          2             2            4        ...
6   shairbat           3              4          2             2            3
7   pigmybat           3              4          2             2            3
```

- We remove the first column (the animal names) and center the data:

```
> X = as.matrix(mammals[,-1])
> X = scale(X,center=TRUE,scale=FALSE)
```

Mammal teeth example

- Unfortunately, using the *cvxclustr* package is a little involved, so have to use the following wrapper. Clustering requires you to specify λ, ϕ and K

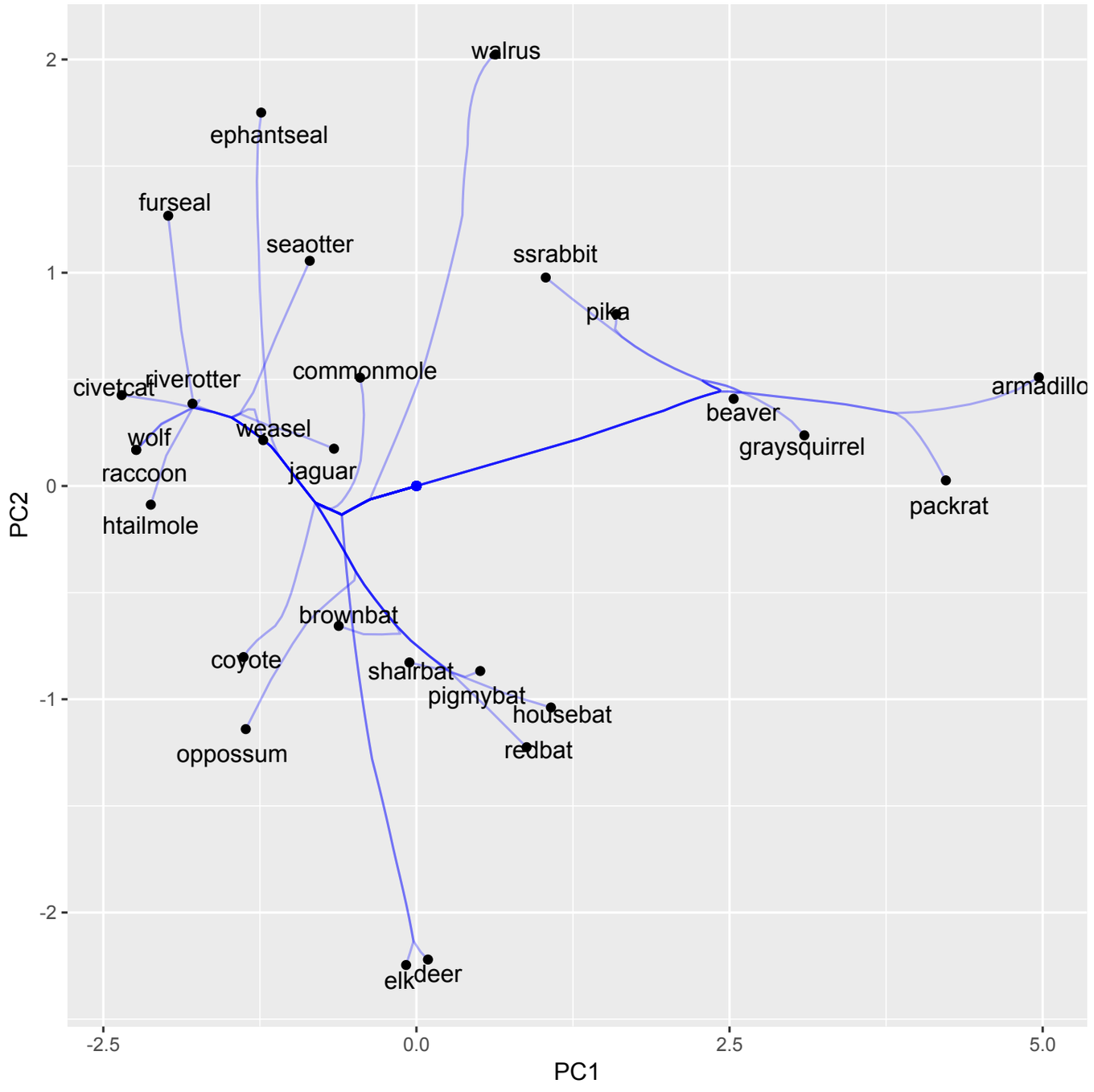
```
cvxclustr_wrapper = function(X,lambda,phi,K,type=2) {
        # default: type=2 is l2-norm penalty term
        # also allowed: type=1 is l1-norm
        X = t(X)
        n = ncol(X)
        w = kernel_weights(X,phi)
        w = knn_weights(w,K,n)
        nu = AMA_step_size(w,n)
        cp = cvxclust_path_ama(X,w,lambda,nu=nu,type=type)
        A = create_adjacency(cp$V[[1]],w,n)
        return(find_clusters(A))
}
```

Mammal teeth example

```
> cvxclust = cvxclustr_wrapper(X,lambda=10,phi=0.5,K=5)
> cvxclust

$cluster
 [1] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 3 1 4 4

$size
[1] 18  6  1  2

mammals[cvxclust$cluster==4,1]

[1] elk  deer

mammals[cvxclust$cluster==3,1]

[1] walrus
```

Mammal teeth example

- With a bit of work (omitted!) we can graphically construct the clustering as λ varies:



Summary

- The question of unsupervised learning is "what we can discover in X?"
- Clustering or cluster analysis is the task of finding subgroups of similar observations in the data
  - discussed K-means, hierarchical clustering and convex clustering
- **K-means** requires us to choose the number of clusters in advance and is usually based on squared Euclidean distance between observations
- **Hierarchical clustering** leads to a tree-based representation of data;
  - have to specify a dissimilarity measure and a linkage method to extend the dissimilarity measure to clusters
- **Convex clustering** includes a penalty term that penalises the distance between cluster centroids, in effect allowing us to tune the number of clusters. However, weights that control the sensitivity to the local density of data are also important
- Clustering is a starting point for further analysis