Unsupervised Learning

Eltecon Data Science Course by Emarsys

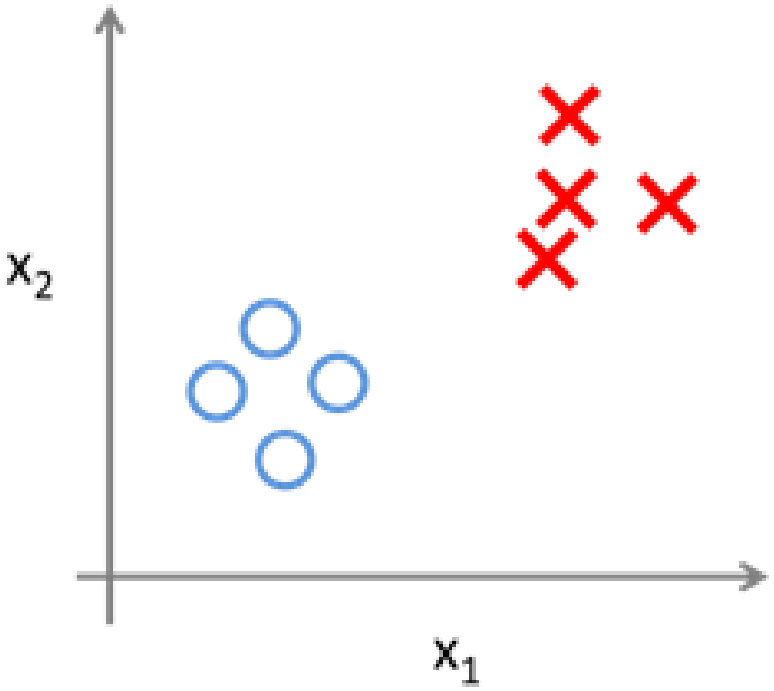
János Divényi

December 2, 2020

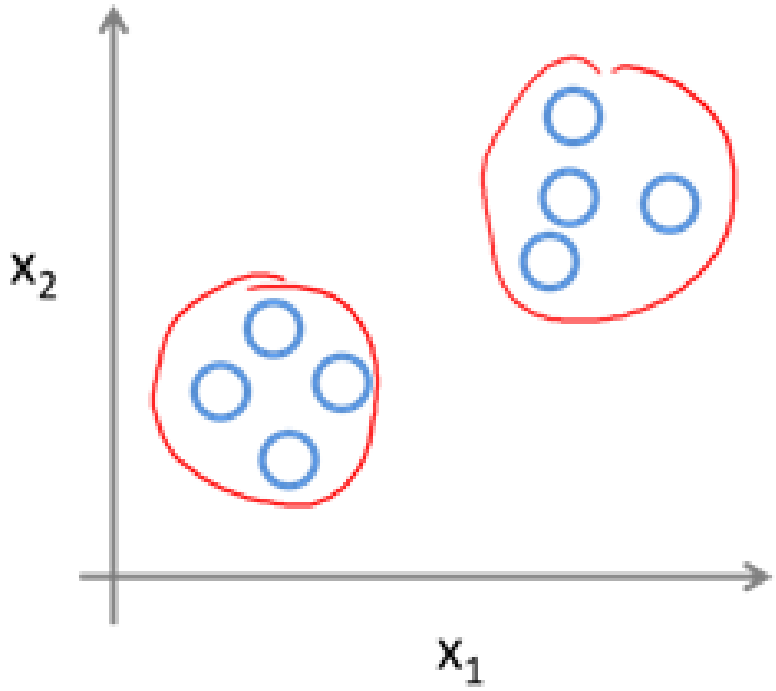
Section 1

Supervised vs unsupervised learning

What is the difference between Supervised and Unsupervised learning?



Supervised Learning



Unsupervised Learning

Supervised learning

- For each observation of the predictor measurements ($\mathbf{X}$) there is an associated response measurement (Y).
- The goal is to fit a model that predicts the amount or the label of the response.
- E.g. linear regression, logistic regression, classification etc.

Unsupervised learning

- We observe measurements ($\mathbf{X}$), but no associated response variable (Y), so we cannot fit any regressions.
- The goal is to find relationship or structure among the measurements.

Goals of unsupervised learning

- Find patterns in the features of the data by dimensionality reduction
  - Ex 1. instead of using both humidity and rainfall in a classification problem, they can be collapsed into just one underlying feature, since both of them are strongly correlated
- Find homogenous subgroups (clusters) within a population
  - Ex 1. segmenting consumers based on demographics and purchasing history
  - Ex 2. find similar movies based on features of each movie and reviews of the movies

Section 2

Dimensionality reduction with PCA

Dimensionality reduction with PCA

- *Principal components* allow us to summarize a large set of correlated variables with a smaller number of representative variables that collectively explain most of the variablility in the original set.
- *Principal component analysis* (PCA) is simply reducing the number of variables of a data set, while preserving as much information as possible.
- Reducing the number of variables comes at the expense of accuracy, so with PCA we trade a little accuracy for simplicity.

What are principal components?

- Try to visualize n observations with measurements of p features by two-dimensional scatterplots (with $p = 10$ there are 45 plots!).
- Instead we'd like to find a low-dimensional representation of the data that captures most of the information.
- Imagine that each of the n observations lives in a p-dimensional space, but not all of these dimensions are equally *interesting*.
- Interesting is measured by the amount that the observations vary along each dimension.
- Each of the dimensions (principal components) found by PCA is a linear combination of the p features.

PCA in R

```
USArrests <- as.data.table(USArrests)
head(USArrests)
```

```
##     Murder Assault UrbanPop  Rape
##      <num>   <int>    <int> <num>
## 1:    13.2     236       58  21.2
## 2:    10.0     263       48  44.5
## 3:     8.1     294       80  31.0
## 4:     8.8     190       50  19.5
## 5:     9.0     276       91  40.6
## 6:     7.9     204       78  38.7
```

PCA in R

```
USArrests[, lapply(.SD, mean)]
```

```
##    Murder Assault UrbanPop  Rape
##     <num>   <num>    <num> <num>
## 1:  7.788   170.8    65.54 21.23
```

```
USArrests[, lapply(.SD, var)]
```

```
##    Murder Assault UrbanPop  Rape
##     <num>   <num>    <num> <num>
## 1:  18.97    6945    209.5 87.73
```

PCA in R

```
pca_output <- prcomp(USArrests, scale = TRUE)
```

```
pca_output$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
pca_output$scale
```

```
##   Murder  Assault UrbanPop     Rape
##    4.356   83.338   14.475    9.366
```

- `center` and `scale` are the *means* and *standard deviations* of the variables that were used for scaling prior to implementing PCA

PCA in R

```
pca_output$rotation
```

```
##                PC1     PC2     PC3      PC4
## Murder   -0.5359  0.4182 -0.3412  0.64923
## Assault  -0.5832  0.1880 -0.2681 -0.74341
## UrbanPop -0.2782 -0.8728 -0.3780  0.13388
## Rape     -0.5434 -0.1673  0.8178  0.08902
```

- `rotation` is the matrix whose columns contain the weights (loadings) for the linear feature combinations of the principal components (mathematically, they are the eigenvectors)

PCA in R

```
dim(pca_output$x)
```

```
## [1] 50  4
```

```
head(pca_output$x)
```

```
##             PC1     PC2      PC3      PC4
## [1,] -0.9757  1.1220 -0.43980  0.15470
## [2,] -1.9305  1.0624  2.01950 -0.43418
## [3,] -1.7454 -0.7385  0.05423 -0.82626
## [4,]  0.1400  1.1085  0.11342 -0.18097
## [5,] -2.4986 -1.5274  0.59254 -0.33856
## [6,] -1.4993 -0.9776  1.08400  0.00145
```

- matrix `x` has as its columns the principal component score vectors

PCA in R

```
pca_var <- pca_output$sdev^2
pca_var
```

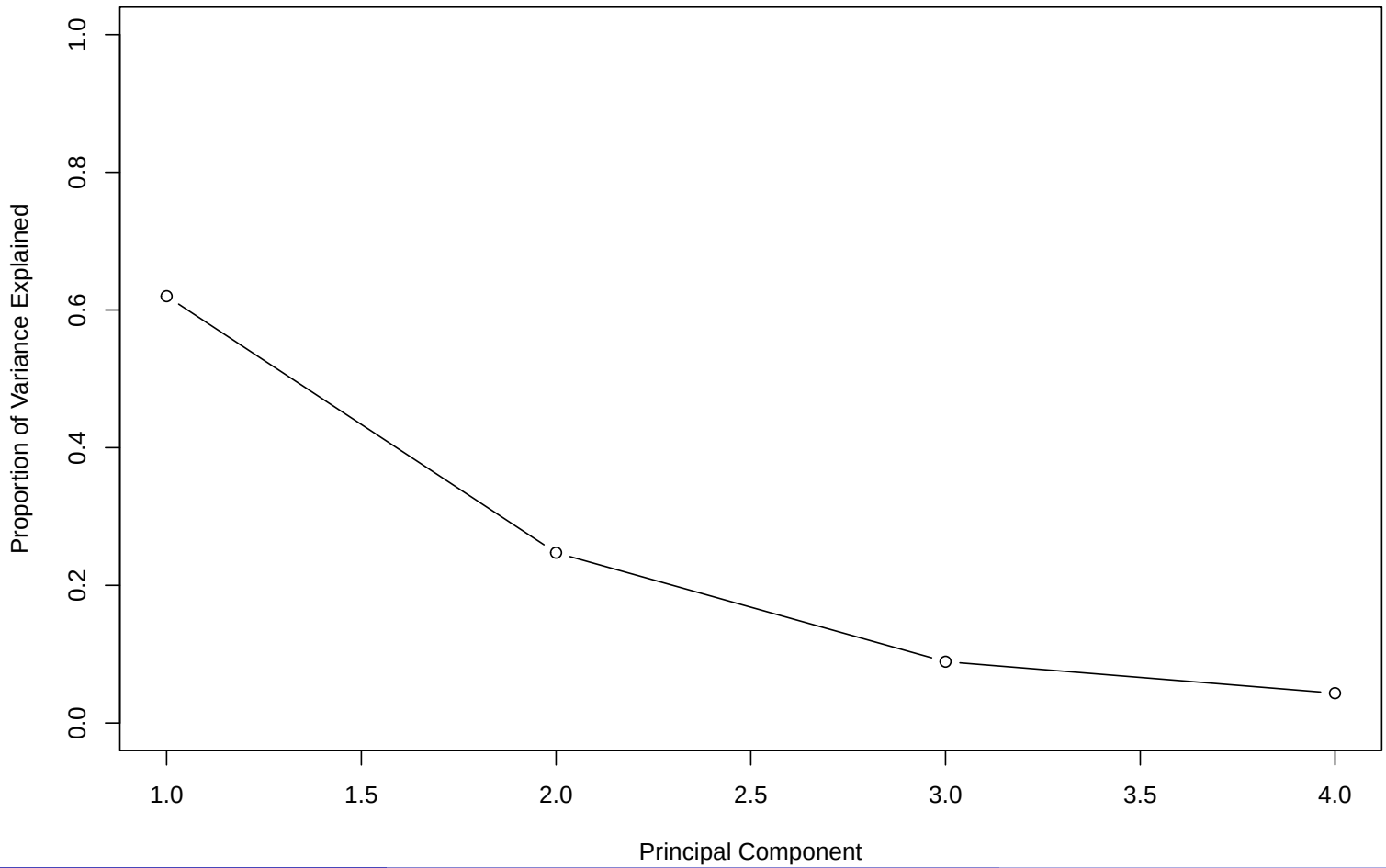
```
## [1] 2.4802 0.9898 0.3566 0.1734
```

```
pca_var_explained <- pca_var / sum(pca_var)
pca_var_explained
```

```
## [1] 0.62006 0.24744 0.08914 0.04336
```

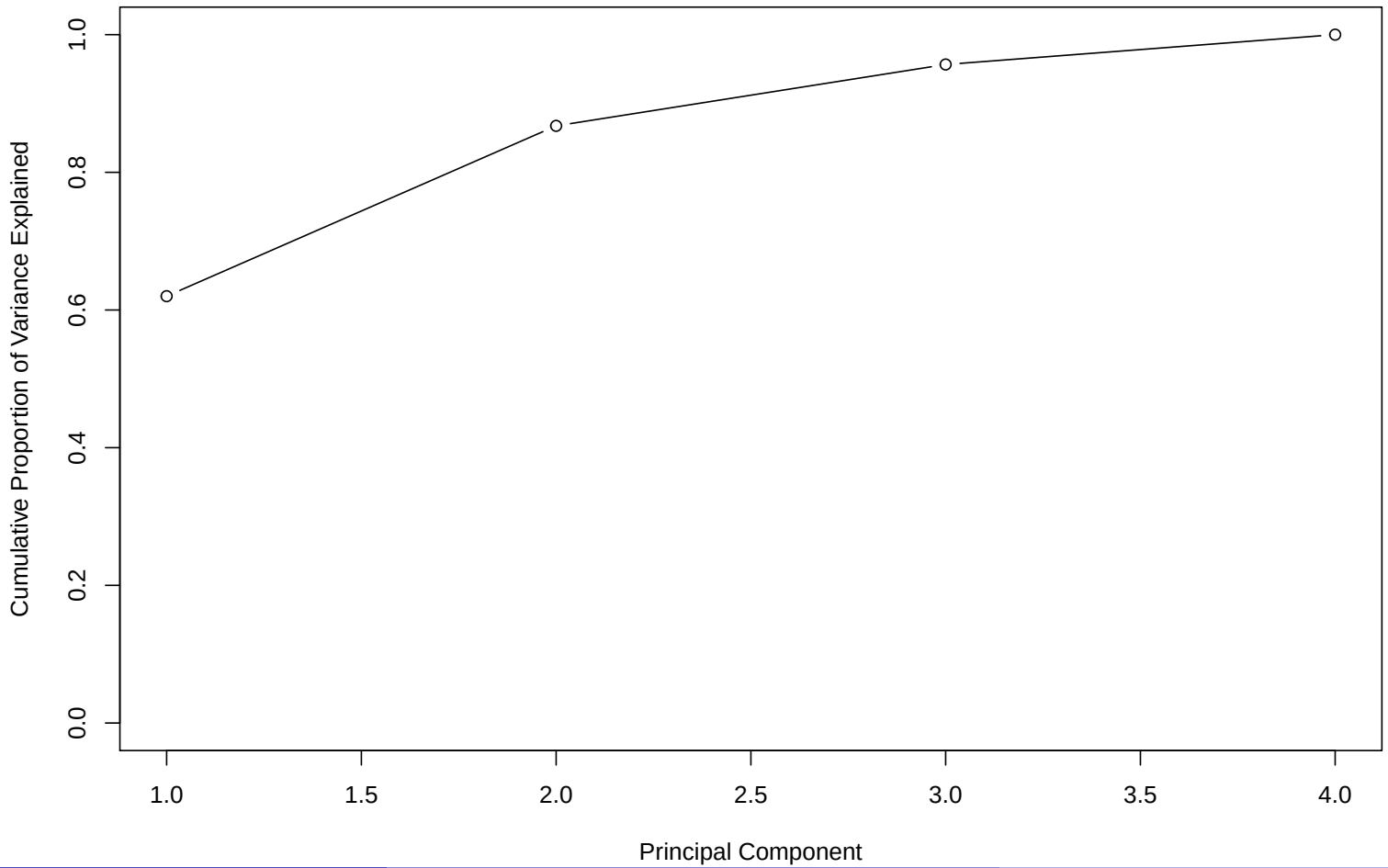
PCA in R

```
plot(pca_var_explained, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```



PCA in R

```
plot(cumsum(pca_var_explained), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```



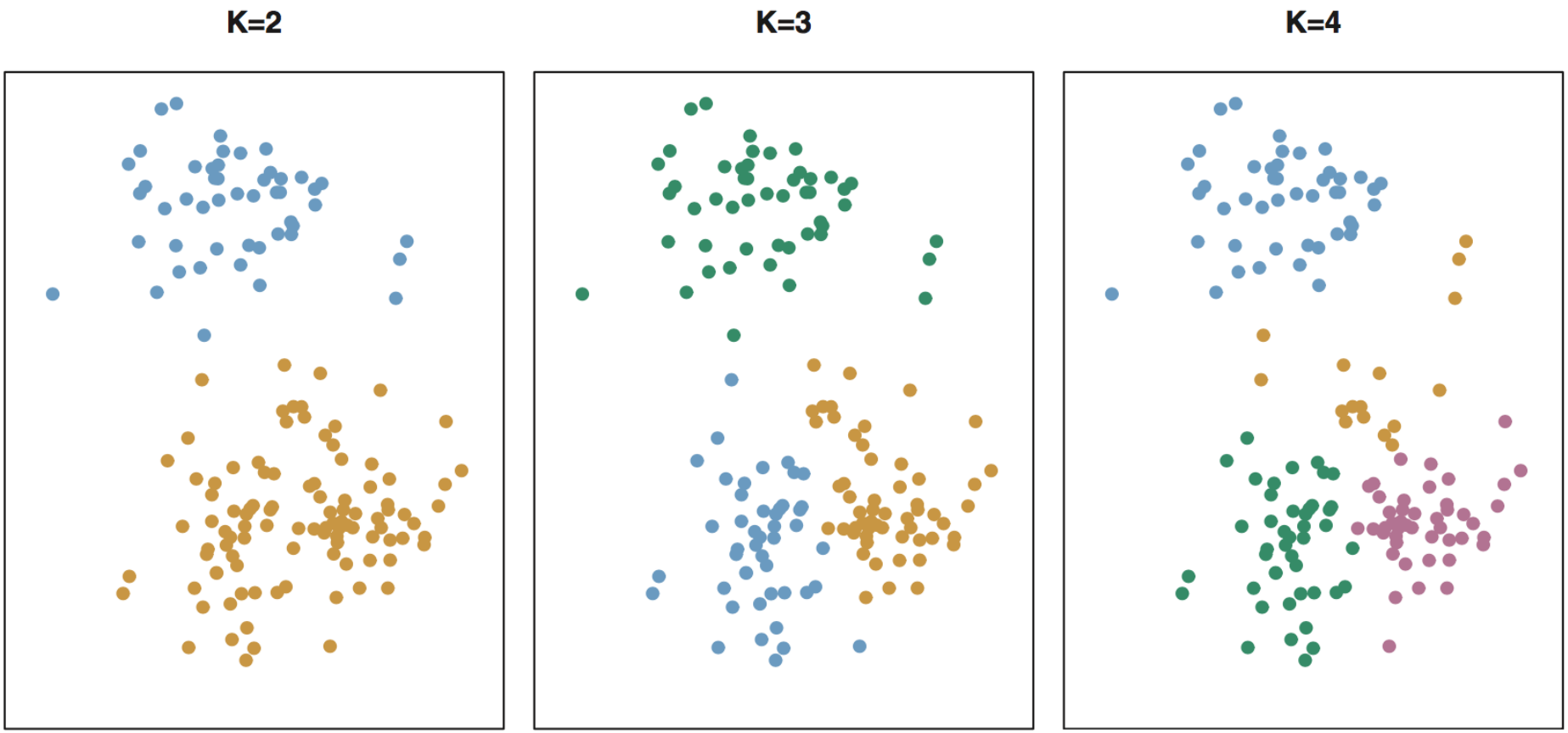
Section 3

K-means clustering

K-means theory

- Clustering the observations of a data set means partitioning them into groups so that observations within each group are quite *similar*, while observations in different groups are quite *different* from each other.
- Each observation should belong to exactly one cluster.
- To perform K-means clustering, we 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.

K-means clustering with different values of K



K-means optimization problem

- A *good* clustering is one for which the *within-cluster variation* is as small as possible.
- If the within-cluster variation for cluster C_k is a measure $W(C_k)$, then we want to solve:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

- The most common choice of measure is the *squared Euclidean distance*:

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

- Thus the optimization problem is:

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

K-means algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) 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.
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

K-means algorithm (cont.)

- The results will depend on the initial (random) cluster assignment in Step 1, thus we need to run the algorithm multiple times from different random initial configurations.
- We select the *best* solution, for which the objective is the smallest.

K-means algorithm explained

- Notice that:

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

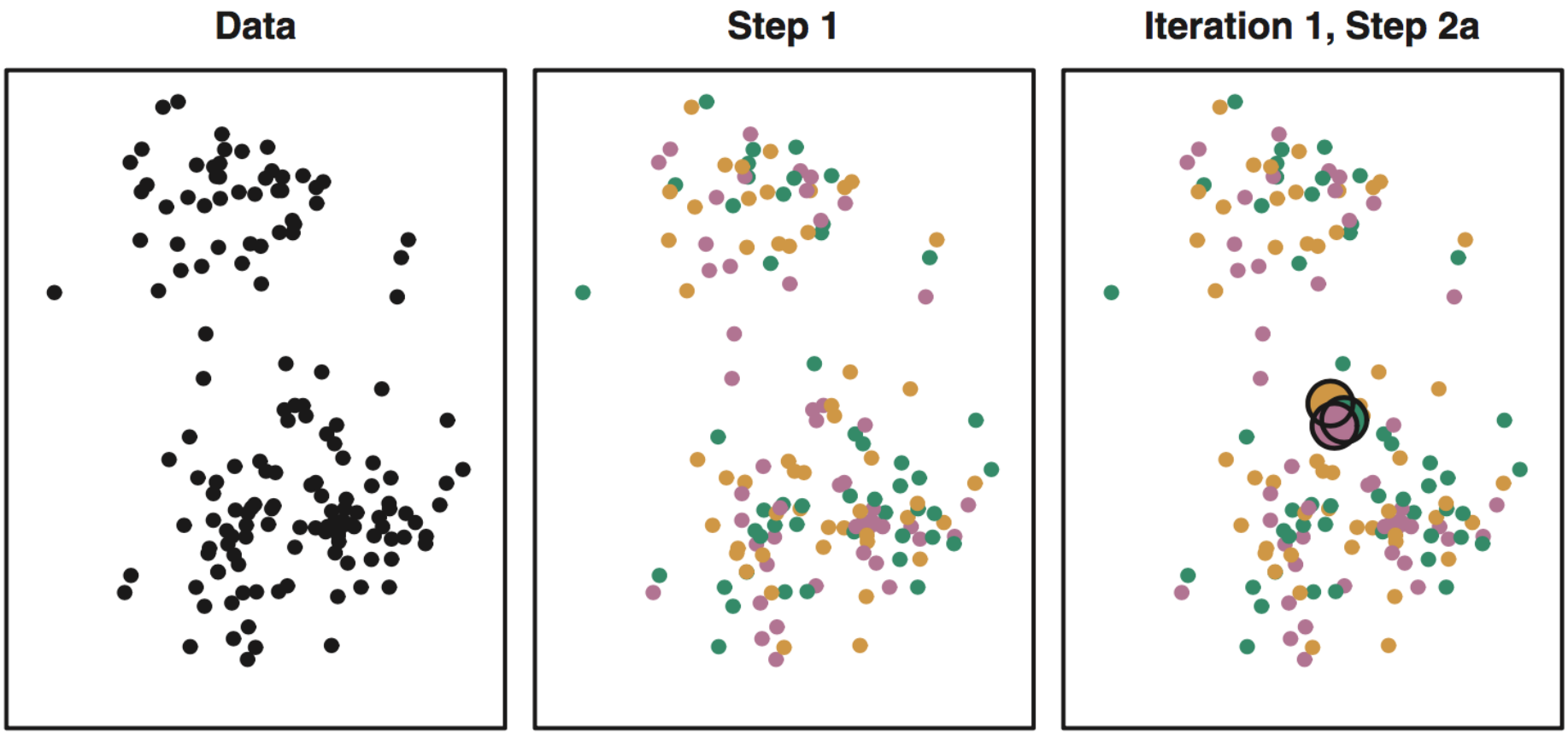
  where

$$\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$$

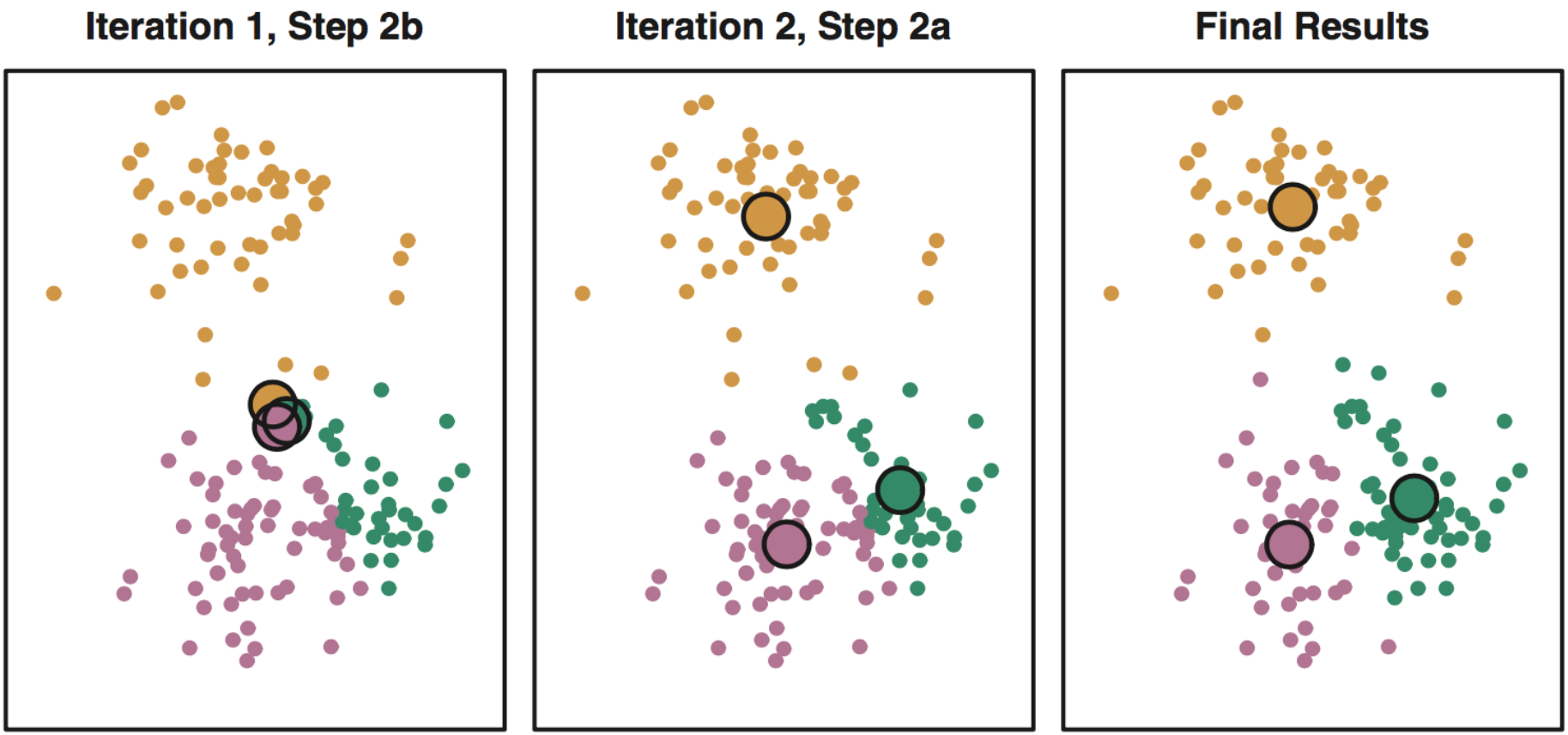
  is the mean for feature j in cluster C_k

- The above shows that in Step 2(a) the cluster means for each feature are the constants that minimize the sum-of-squared deviations, and by reallocation in Step 2(b) we can only improve.
- When the result no longer changes, a *local optimum* has been reached.

K-means clustering progression



K-means clustering progression (cont.)



K-means clustering with K=2

```
set.seed(2)
x <- data.table(a = rnorm(50), b = rnorm(50))
x[1:25, `:=`(a = a + 2, b = b - 3)]
```

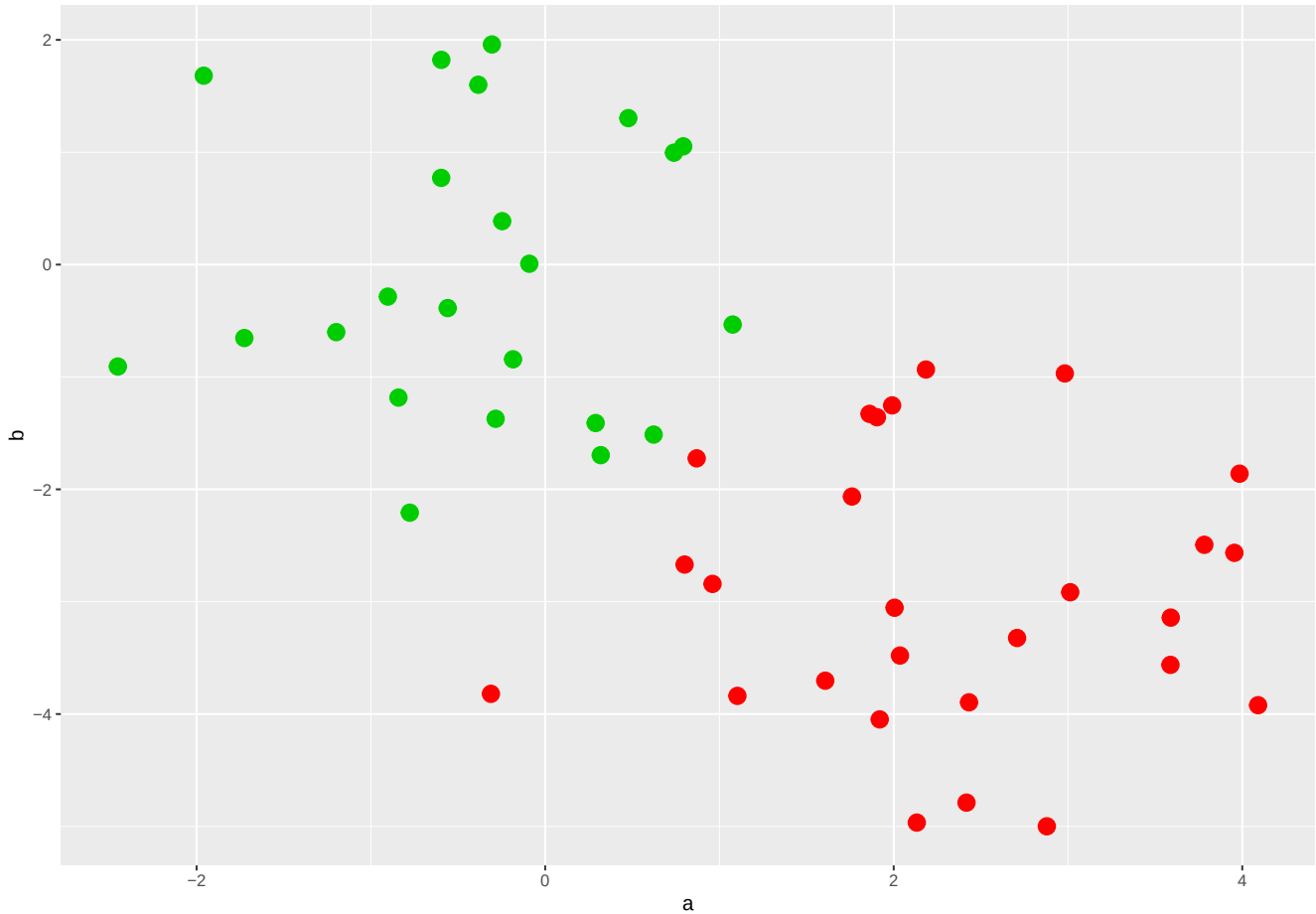
```
km_output <- kmeans(x, centers = 2, nstart = 20)
```

```
km_output$cluster
```

```
##  [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
```

K-means clustering with K=2

```
ggplot(x, aes(x = a, y = b)) +
    geom_point(colour = (km_output$cluster + 1), size = 4)
```

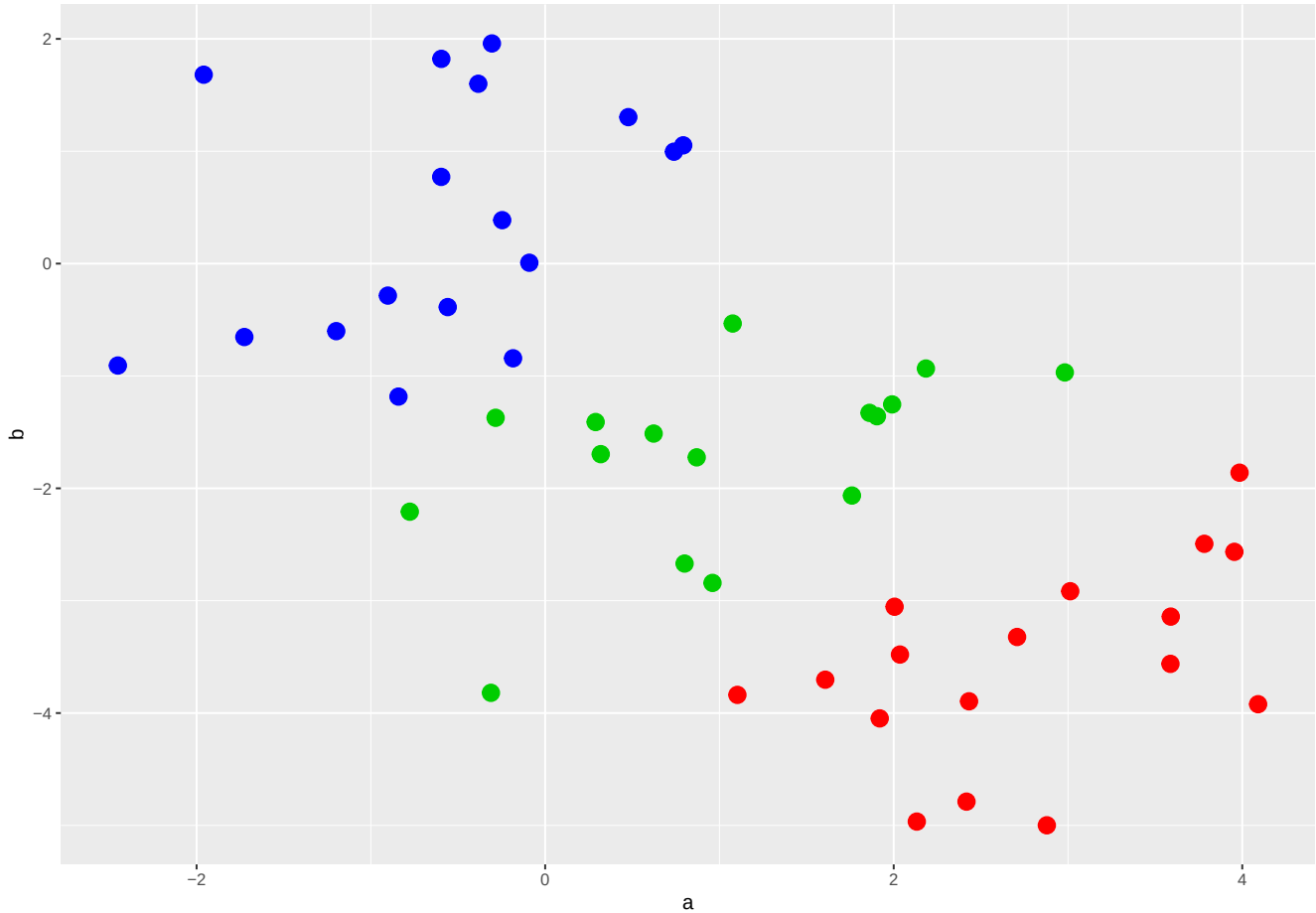


K-means clustering with K=3

```
set.seed(4)
km_output <- kmeans(x, centers = 3, nstart = 20)
```

K-means clustering with K=3

```
ggplot(x, aes(x = a, y = b)) +
    geom_point(colour = (km_output$cluster+1), size = 4)
```



Different nstart initial cluster assignments result different total within-cluster sum of squares

```
set.seed(3)
km_output <- kmeans(x, centers = 3, nstart = 1)
km_output$tot.withinss
```

```
## [1] 94.76
```

```
km_output <- kmeans(x, centers = 3, nstart = 5)
km_output$tot.withinss
```

```
## [1] 83.18
```

```
km_output <- kmeans(x, centers = 3, nstart = 10)
km_output$tot.withinss
```

```
## [1] 83.18
```

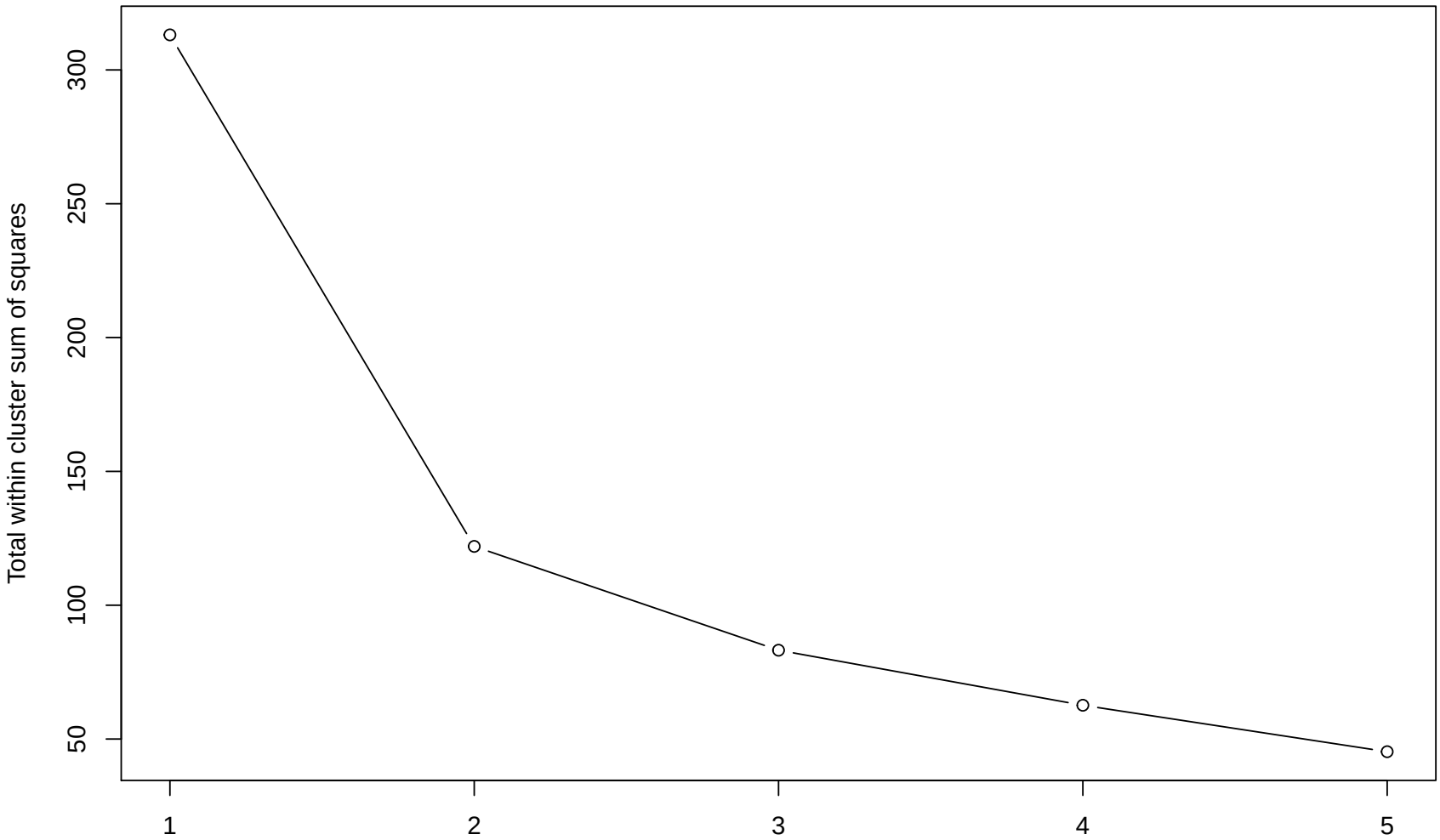
How to determine the number of clusters?

1. Run K-means with K=1, K=2, . . . , K=n.
2. Record total within-cluster sum of squares for each value of K.
3. Choose K at the *elbow* position.

```
ks <- 1:5
tot_within_ss <- sapply(ks, function(k) {
    km_output <- kmeans(x, k, nstart = 20)
    km_output$tot.withinss
})
```

How to determine the number of clusters?

```
plot(ks, tot_within_ss, type = "b", xlab = "Values of K",
     ylab = "Total within cluster sum of squares")
```



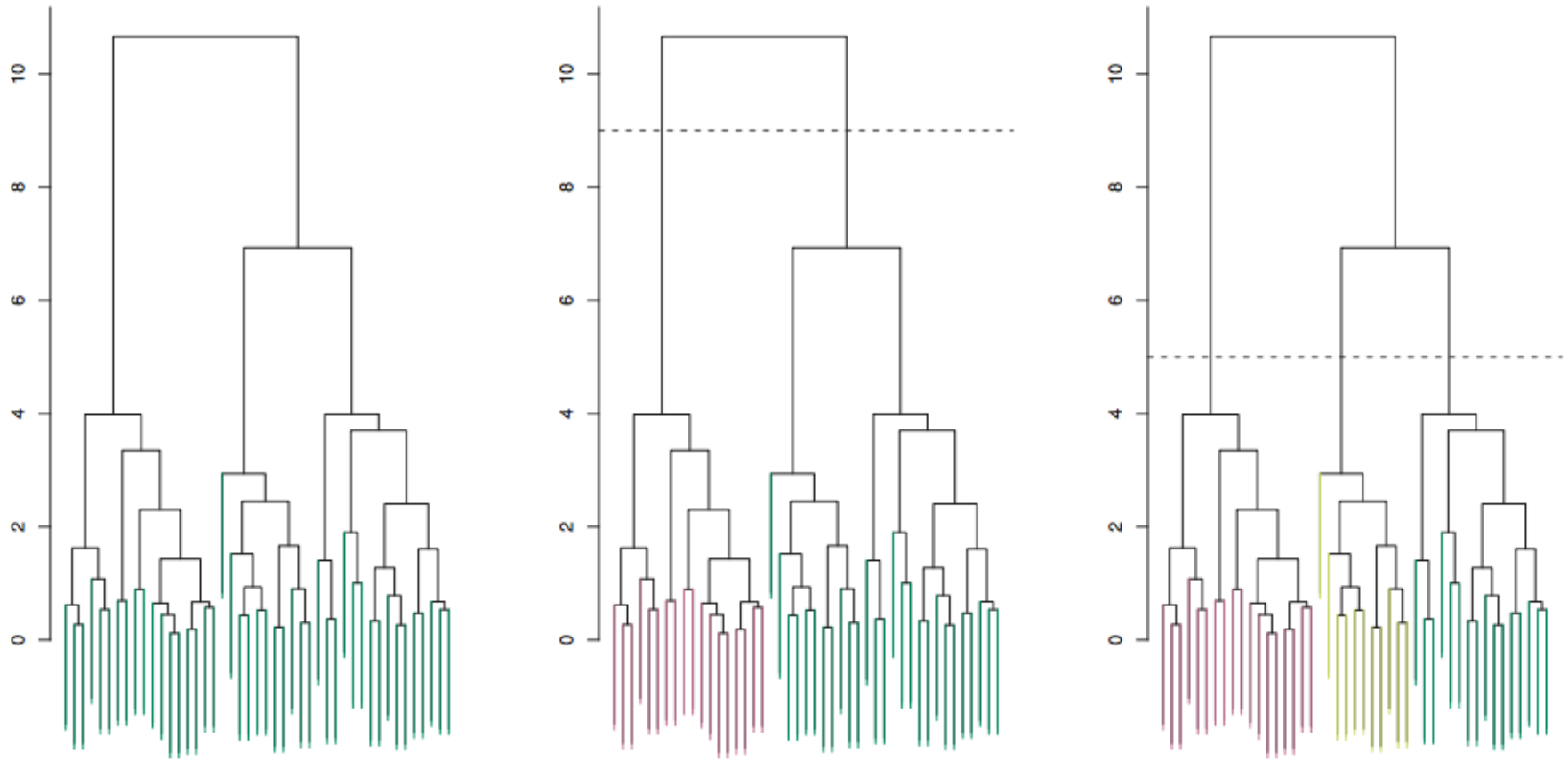
Section 4

Hierarchical clustering

Hierarchical clustering

- Hierarchical clustering does not require choosing a particular K number of clusters.
- It results in a tree-based representation of the observations, called a *dendogram*.
- We focus on *bottom-op* or *agglomerative* clustering (vs *top-down* or *divisive*).

The dendogram



The dendogram

- Each *leaf* is an observation, as we move up the tree, leafs begin to *fuse* into branches based on their *similarity*.
- The height of fusion (on the vertical axis) indicates how different the observations are.
- Clusters are defined by cuting the dendogram horizontally, although where to make the cut is not so obvious.
- *Hierarchical* refers to that clusters obtained by cutting the tree at a given height are necessarily nested within the clusters obtained by cutting higher.

Hierarchical clustering algorithm

- We need to define a *dissimilarity measure* between each pair of observations (most often Euclidean distance).
- Starting from the bottom, each observation is treated as a separate cluster, then the two most similar are *fused*, next the two most similar clusters are fused, etc., until all observations belon to a cluster and the dendogram is complete.
- Dissimilarity between clusters depend on the selected *linkage* (average and complete linkage are the most preferred ones) and the dissimilarity measure.

Linkage types

- The linkage function tells you how to measure the distance between clusters.
- Single linkage: Minimal intercluster dissimilarity.

$$f = min(d(x, y))$$

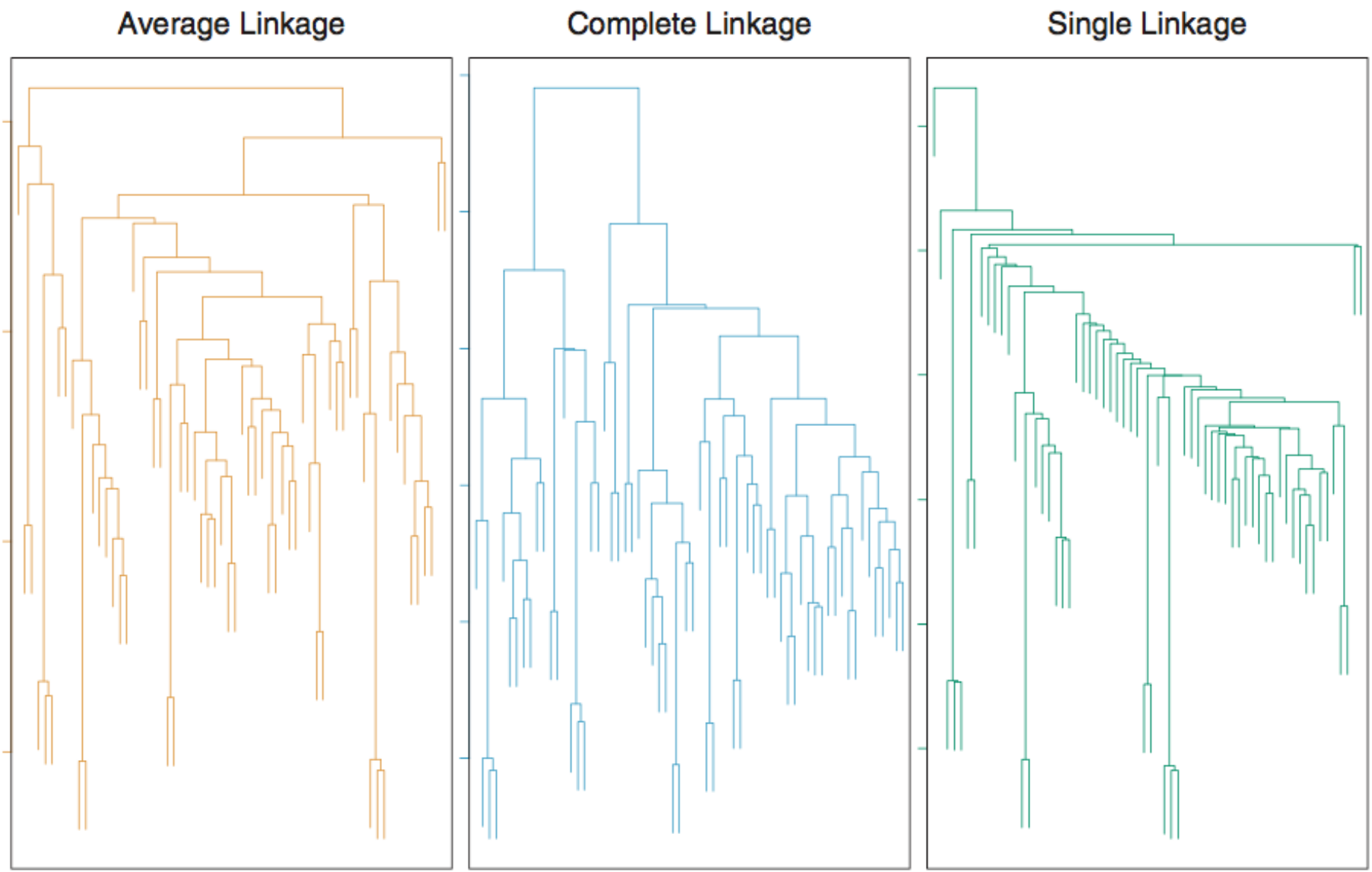
- Complete linkage: Maximal intercluster dissimilarity.

$$f = max(d(x, y))$$

- Average linkage: Mean intercluster dissimilarity.

$$f = average(d(x, y))$$

Clustering with different linkages



Dissimilarity measures

- Euclidean distance is the most common measure used.

$$\sqrt{\sum_i (a_i - b_i)^2}$$

- *Correlation-based distance* is also very useful, e.g. it is used for gene expression.
  - It considers two observations to be similar if their features are highly correlated, even though the observed values may be far apart in terms of Euclidean distance.
  - The distance between two vectors is 0 when they are perfectly correlated.
  - It focuses on the shapes of observation profiles rather than their magnitudes.

Hierachical clustering using different linkage types

```
set.seed(2)
x <- data.table(a = rnorm(50), b = rnorm(50))
x[1:25, `:=`(a = a + 3, b = b - 4)]
```

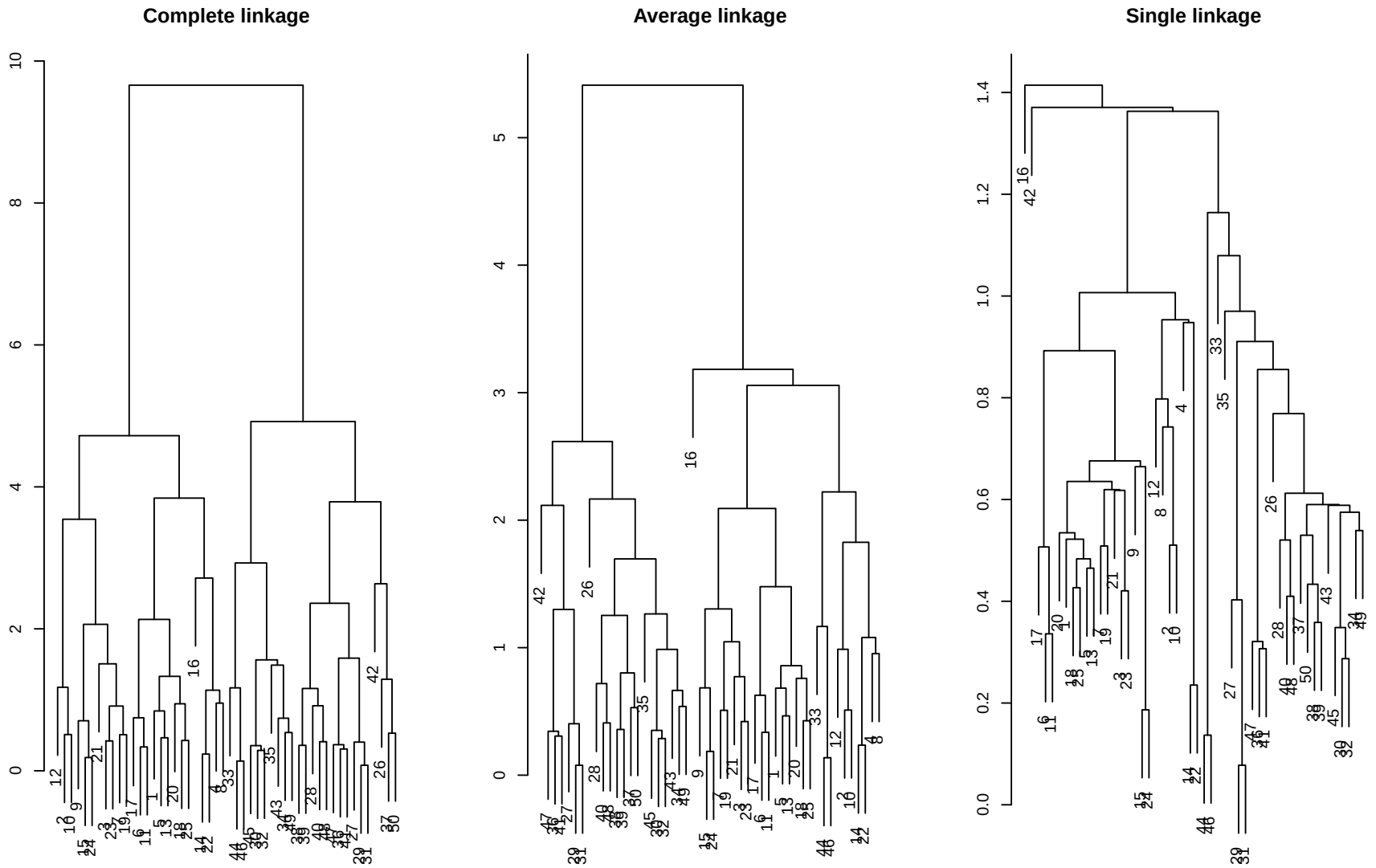
```
hc_complete <- hclust(dist(x), method = "complete")
hc_average <- hclust(dist(x), method = "average")
hc_single <- hclust(dist(x), method = "single")
```

- Compute the inter-observation Euclidean distance matrix using `dist()`
- Set the linkage type in the `method` argument

Preparing the dendogram

```
par(mfrow = c(1, 3))
plot(hc_complete, main = "Complete linkage", xlab = "",
     ylab = "", sub = "")
plot(hc_average, main = "Average linkage", xlab = "",
     ylab = "", sub = "")
plot(hc_single, main = "Single linkage", xlab = "",
     ylab = "", sub = "")
```

Preparing the dendogram



Determining cluster labels with `cutree()` by selecting number of clusters (`k`)

```
cutree(hc_complete, k = 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 2 2
```

```
cutree(hc_average, k = 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 2 2
```

```
cutree(hc_single, k = 2)
```

```
##   [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
```

Determining cluster labels with `cutree()` by selecting maximum distance (`h`)

```
cutree(hc_complete, h = 5)
```

```
##   [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
```

```
cutree(hc_average, h = 4)
```

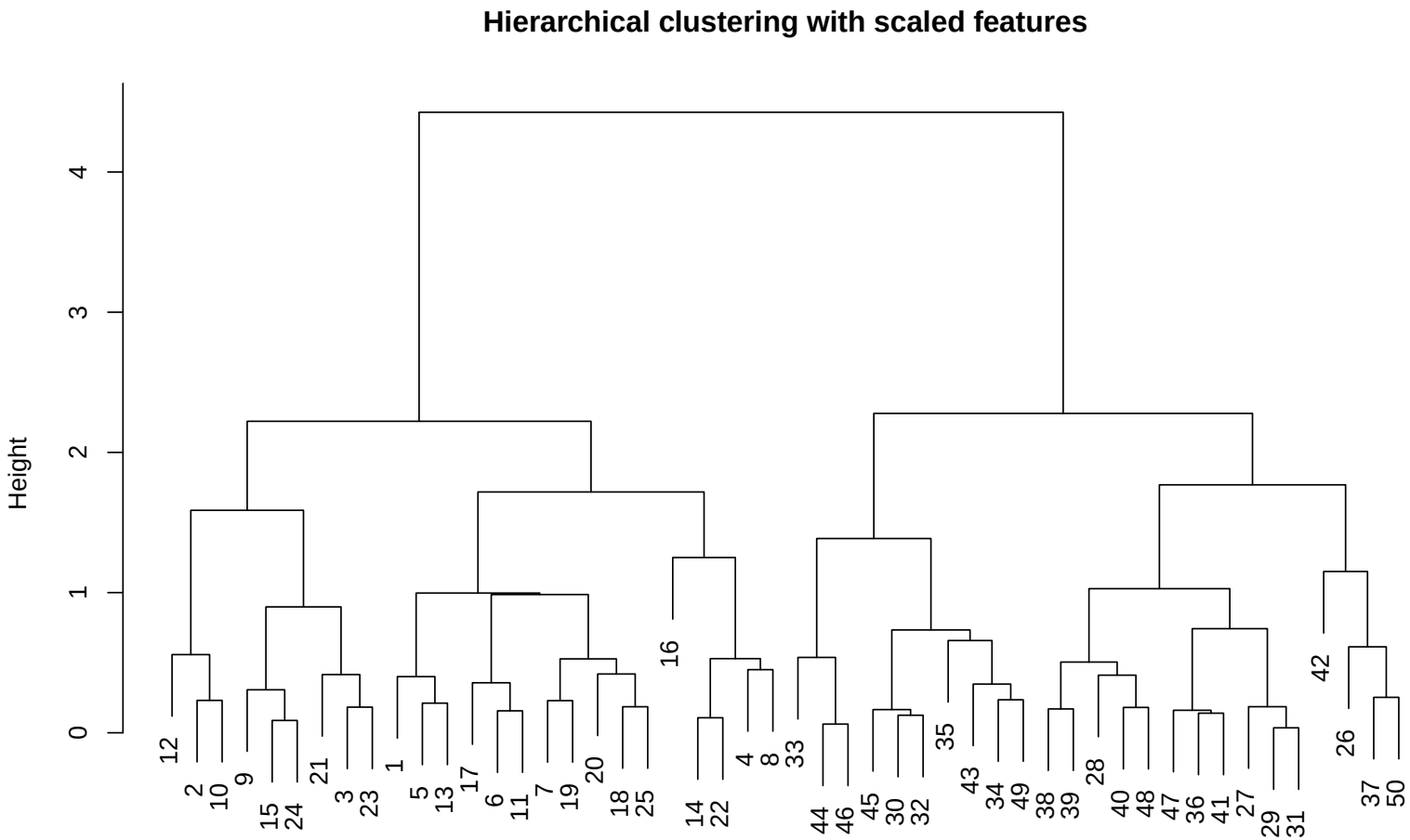
```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
```

```
cutree(hc_single, h = 1.4)
```

```
##   [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
```

Scale variables with `scale()`

```
x_scaled <- scale(x)
plot(hclust(dist(x_scaled), method = "complete"),
     main = "Hierarchical clustering with scaled features")
```

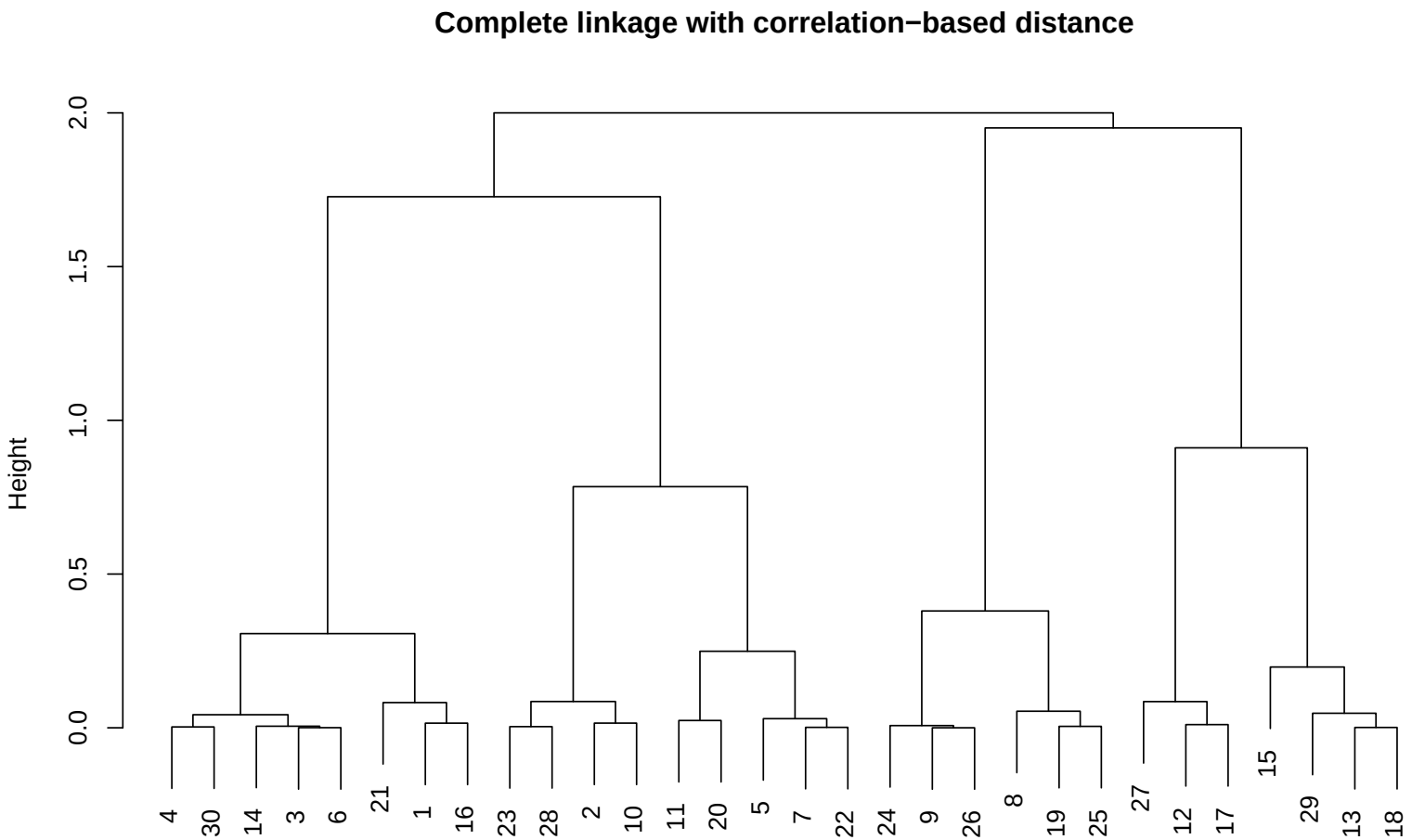


Compute correlation-based distance using `as.dist()`

```
set.seed(5)
x <- matrix(rnorm(30 * 3), ncol = 3)
dd <- as.dist(1 - cor(t(x)))
```

Compute correlation-based distance using as.dist()

```
plot(hclust(dd, method = "complete"),
     main = "Complete linkage with correlation-based distance",
     xlab = "", sub = "")
```



Section 5

Summary

Summary

- Supervised vs unsupervised learning
- PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance.
- Clustering looks to find homogeneous subgroups among the observations.
  - In K-means clustering, we seeks to partition the observations into a pre-specified number of clusters.
  - In hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

Resources

- StatQuest videos:
  - PCA #1
  - PCA #2
  - Hierarchical clustering
  - K-Means clustering
- James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning with Applications in R.*