Topic08.Unsupervised learning

資料科學 Data Science
張家銘 Jia-Ming Chang
政治大學資訊科學系

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

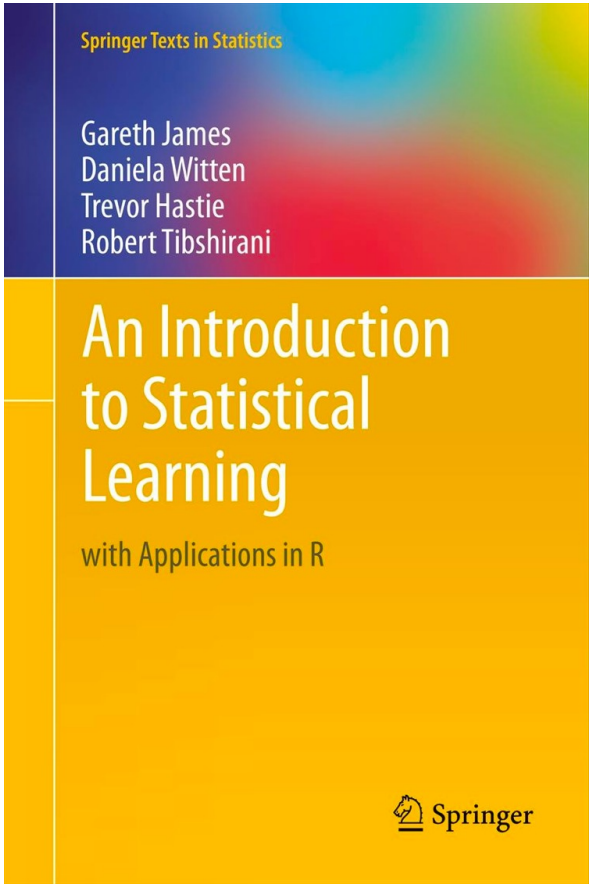


Figure 3.18, *An Introduction to Statistical Learning with Applications in R,* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "Practical Data Science with R (Manning, 2019)"
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

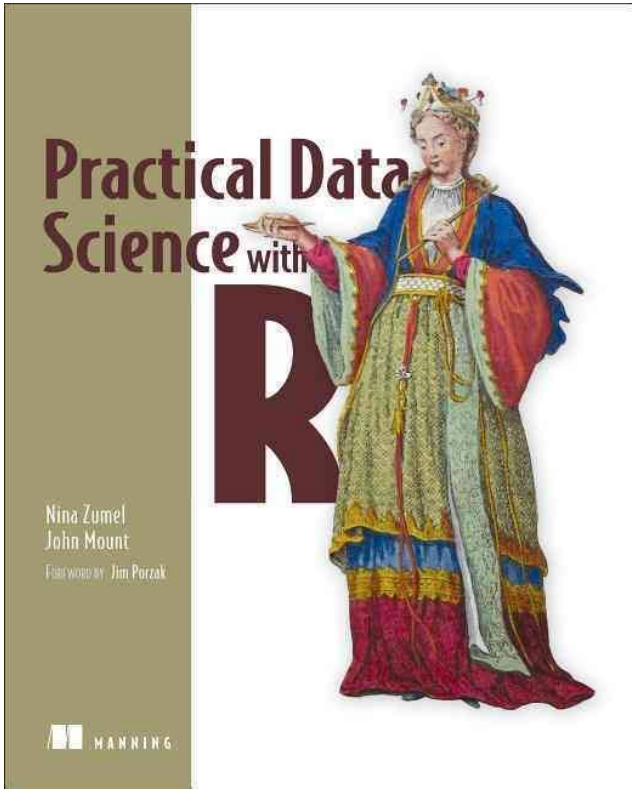
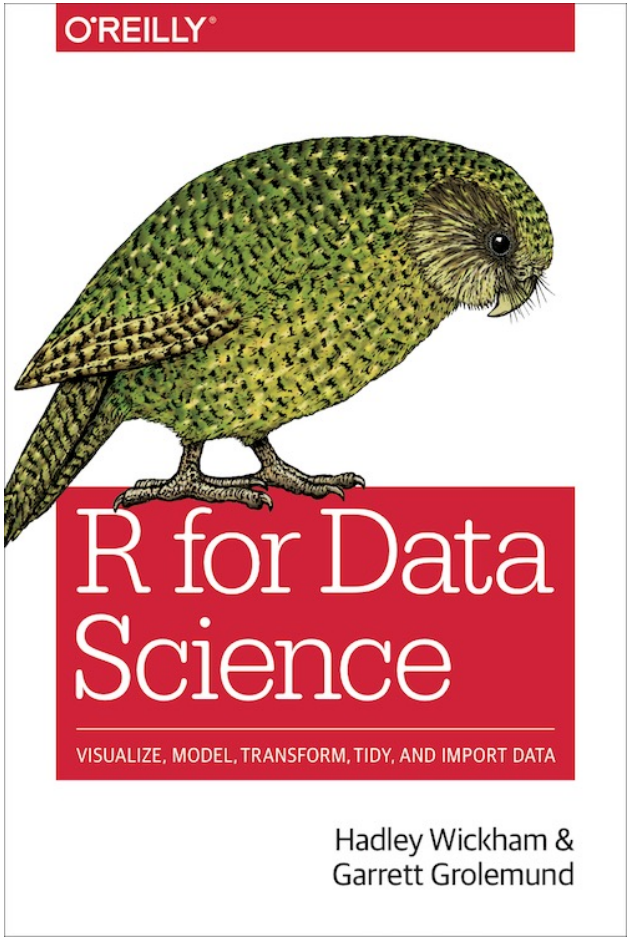


Figure 7.6, *Practical Data Science with R* by Nina Zumel and John Mount

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "R for Data Science" under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.
- The web site of the book
- The credit of individual is indicated in the bottom part.
  - ie.,

R for Data Science by Garrett Grolemund, Hadley Wickham



Recap from the last week

knit the document R Markdown

diamond-sizes.Rmd | simple.md

Knit

```
---
title: "Diamond sizes"
date: 2016-08-25
output: html_document
---

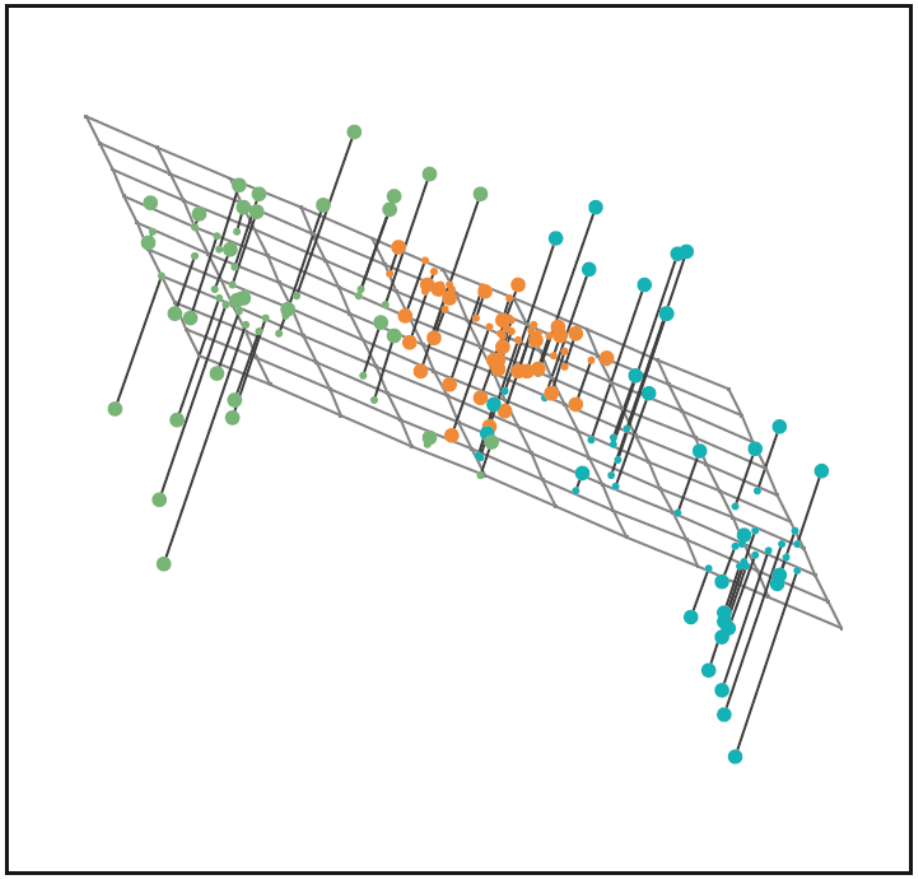
```{r setup, include = FALSE}
library(ggplot2)
library(dplyr)

smaller <- diamonds %>%
  filter(carat <= 2.5)
```

We have data about `r nrow(diamonds)` diamonds. Only
`r nrow(diamonds) - nrow(smaller)` are larger than
2.5 carats. The distribution of the remainder is shown
below:

```{r, echo = FALSE}
smaller %>%
  ggplot(aes(carat)) +
  geom_freqpoly(binwidth = 0.01)
```
```

supervised vs. unsupervised learning

- supervised learning
  - Input: a set of p features $X_1, X_2, \ldots, X_p$, measured on n observations, and a response Y also measured on those same n observations
  - Goal: predict Y using $X_1, X_2, \ldots, X_p$
- unsupervised learning
  - Input: $X_1, X_2, \ldots, X_p$ measured on n observations
  - Output: discover interesting things about the measurements



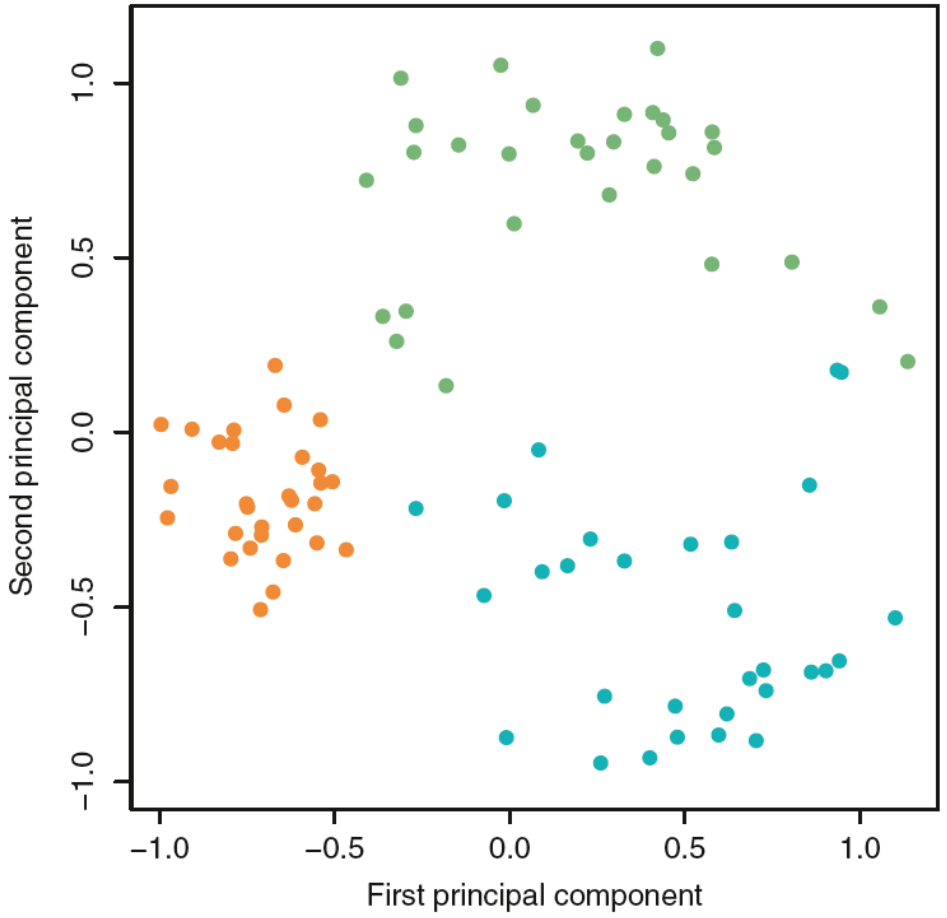
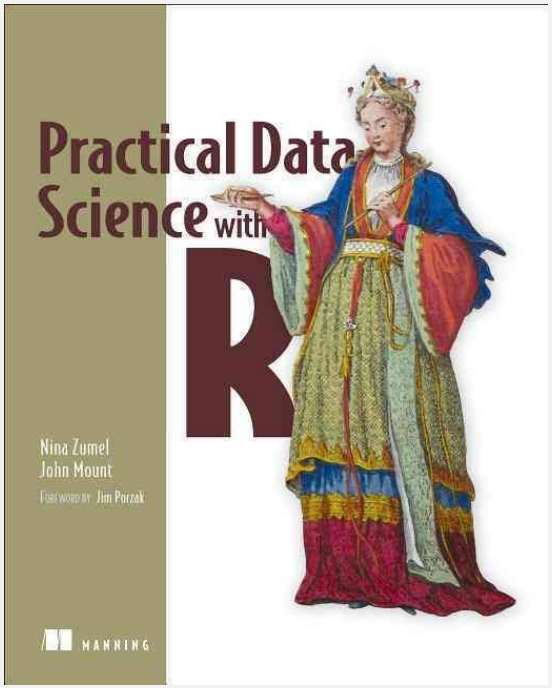
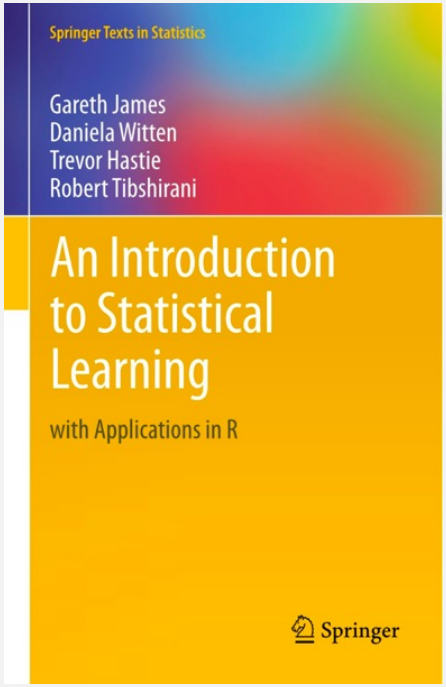


Figure10.2, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Chp. 8: Unsupervised methods





Chp. 10.3 Clustering Methods

Unsupervised methods

- we'll look at methods to discover unknown relationships in data
- two classes of unsupervised methods
  - *Cluster analysis* : finds groups in your data with similar characteristics
    - Hierarchical clustering
    - k-means
  - *Association rule* : mining finds elements or properties in the data that tend to occur together.

Clustering analysis

Unsupervised learning

Clustering

- Refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set
  - *n* observations = tissue samples for patients with breast cancer
  - *p* features = clinical measurements: tumor stage or grade, or gene expression measurements
  - a few different unknown subtypes of breast cancer
- Clustering looks to find homogeneous subgroups among the observations.

Distances

- distance.R
- Euclidean distance (squared Euclidean distance = L2 distance)
  - when all the data is real-valued (quantitative)

```
edist(x,y)<-sqrt((x[1]-y[1])^2+(x[2]-
y[2])^2+...)
```

- Hamming distance
  - categorical variables (male /female , or small /medium /large)

```
hdist(x,y)<-sum((x[1]!=y[1])+(x[2]!= y[2])+ ...)
```

Distances

- Manhattan (city block) distance : L1 distance
  - # of horizontal and vertical units from one point to the other

```
mdist(x,y)<-sum(abs(x[1]-y[1])+abs(x[2]-y[2])
+...)
```

- Cosine similarity

```
dot(x, y)<-sum( x[1]*y[1] + x[2]*y[2] + ... )
cossim(x, y)<-dot(x,
y)/(sqrt(dot(x,x)*dot(y,y)))
1 - 2*acos(cossim(x,y))/pi
```

Euclidean distance vs. correlation-based distance

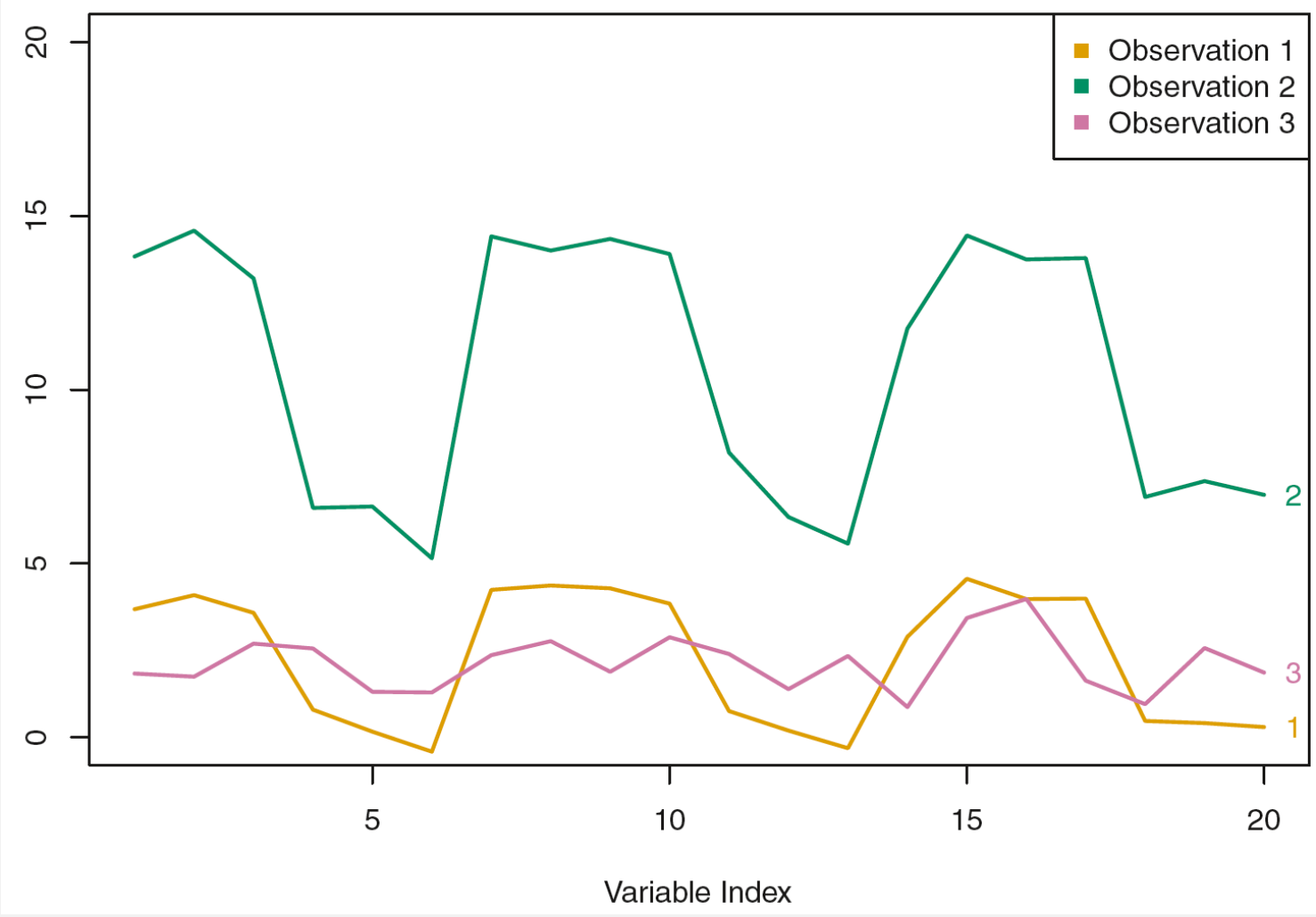


Figure10.13, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

clustering approaches

- *K*-means clustering
  - we seek to partition the observations into a pre-specified number of clusters.
- 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, *dendrogram*
    - allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to *n*.

Prepare data

- a small dataset from 1973 on protein consumption from nine different food groups in 25 countries in Europe => group the countries based on patterns in their protein consumption.
- protein/protein.R

```
protein <- read.table("protein.txt", sep="\t",
header=TRUE)
```

Prepare data

- What should you do first?

```
vars.to.use <- colnames(protein)[-1]
pmatrix <- scale(protein[,vars.to.use])
pcenter <- attr(pmatrix, "scaled:center")
pscale <- attr(pmatrix, "scaled:scale")
```

To Scale or not to scale

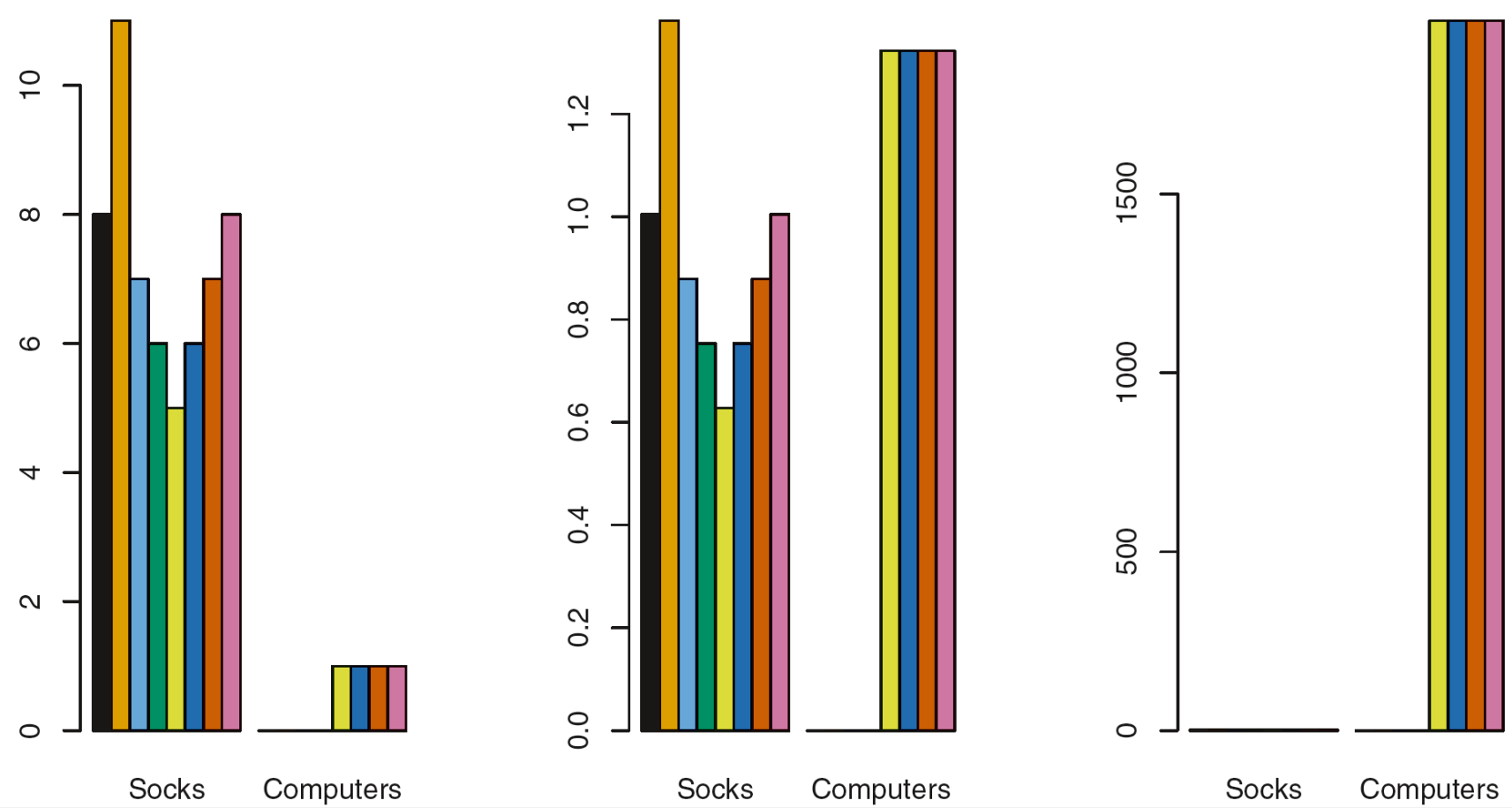


Figure10.14, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Hierarchical clustering

Unsupervised learning

Clustering analysis

Hierarchical clustering

- an alternative approach which does not require that we commit to a particular choice of K
- results in an attractive tree-based representation of the observations, called a *dendrogram*

Interpreting a Dendrogram

the vertical axis, indicates how different the two observations are.

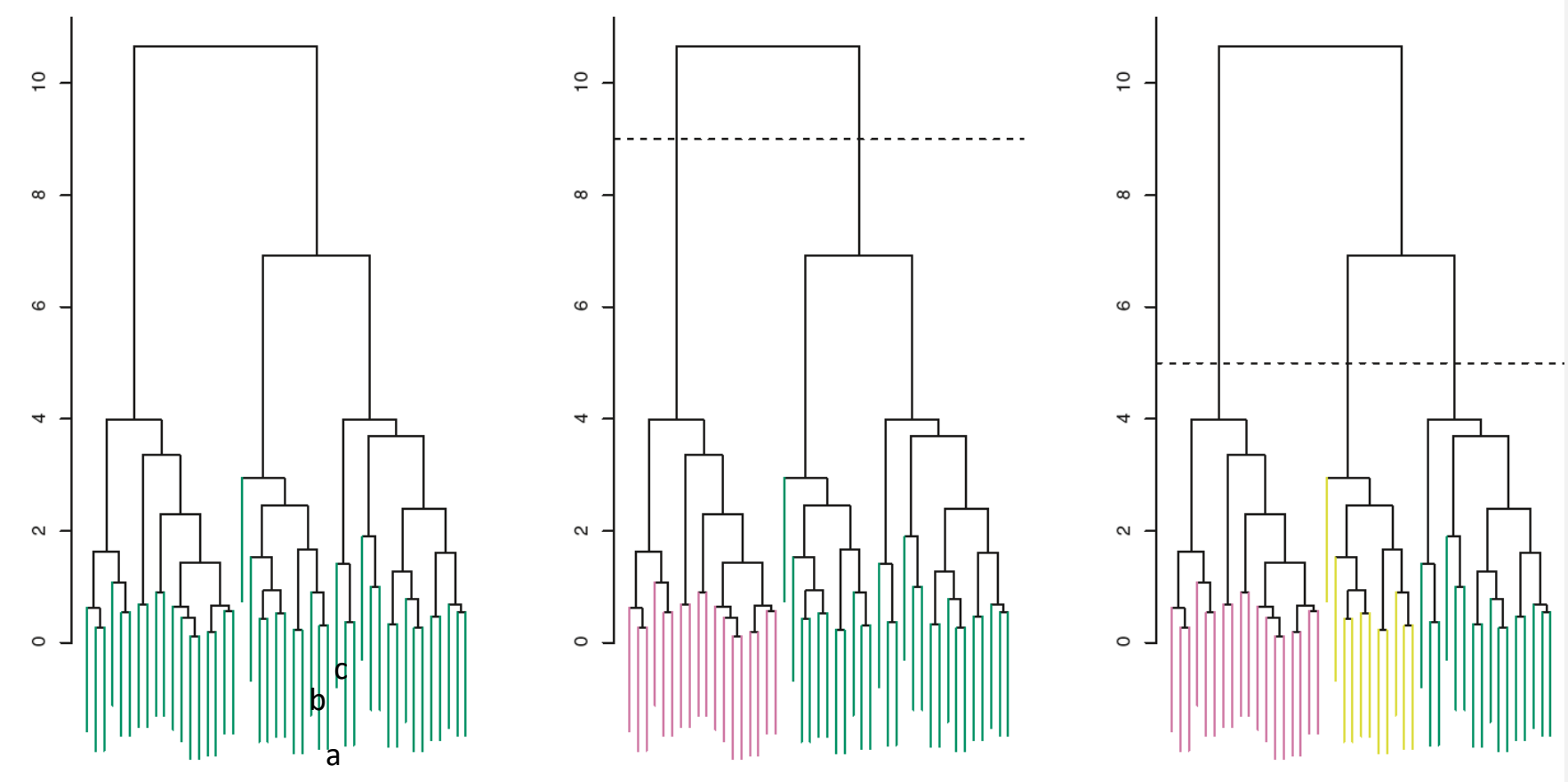


Figure10.9, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Forty-five observations generated in two-dimensional space

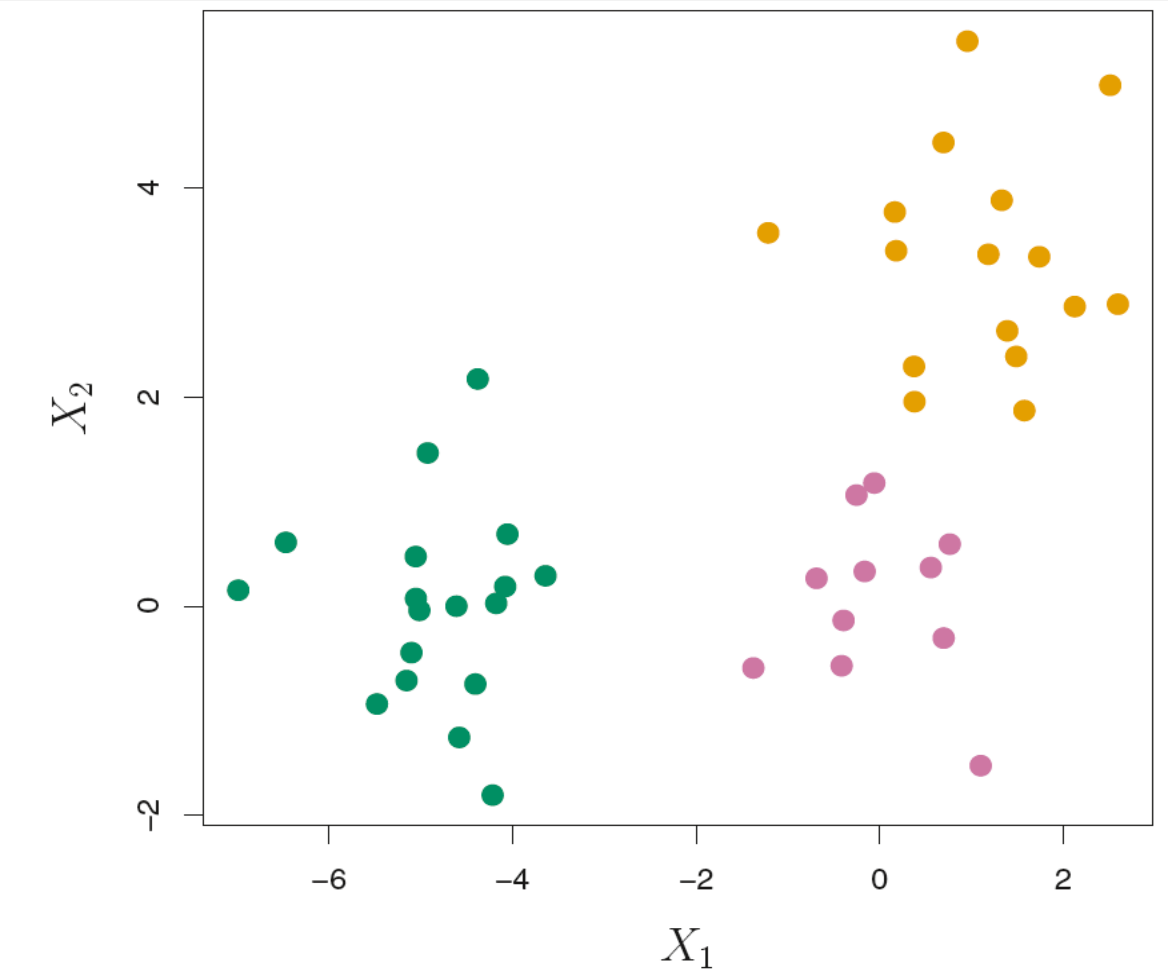
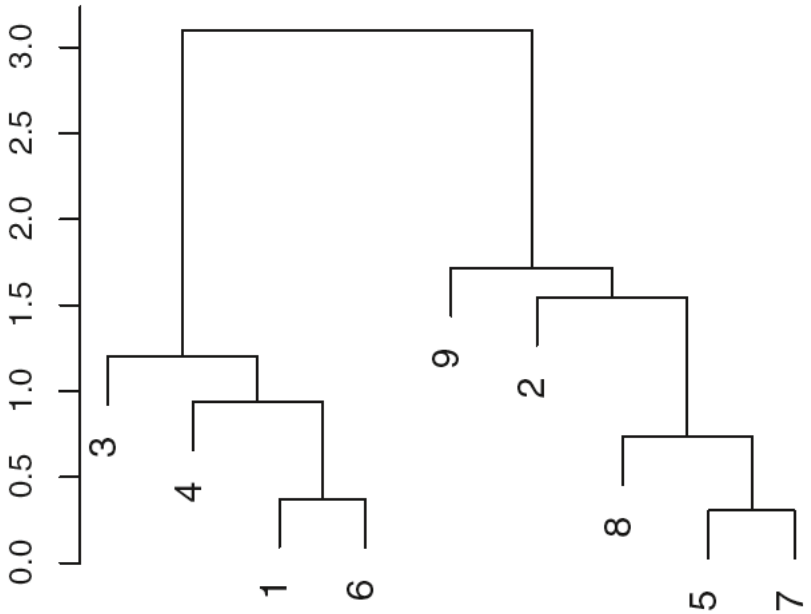


Figure10.8, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Interpreting a Dendrogram

- the similarity of two observations based
  - on the location on the vertical axis where branches containing those two observations first are fused.
  - ~~on their proximity along the horizontal axis the similarity of two observations based~~

Interpreting a Dendrogram



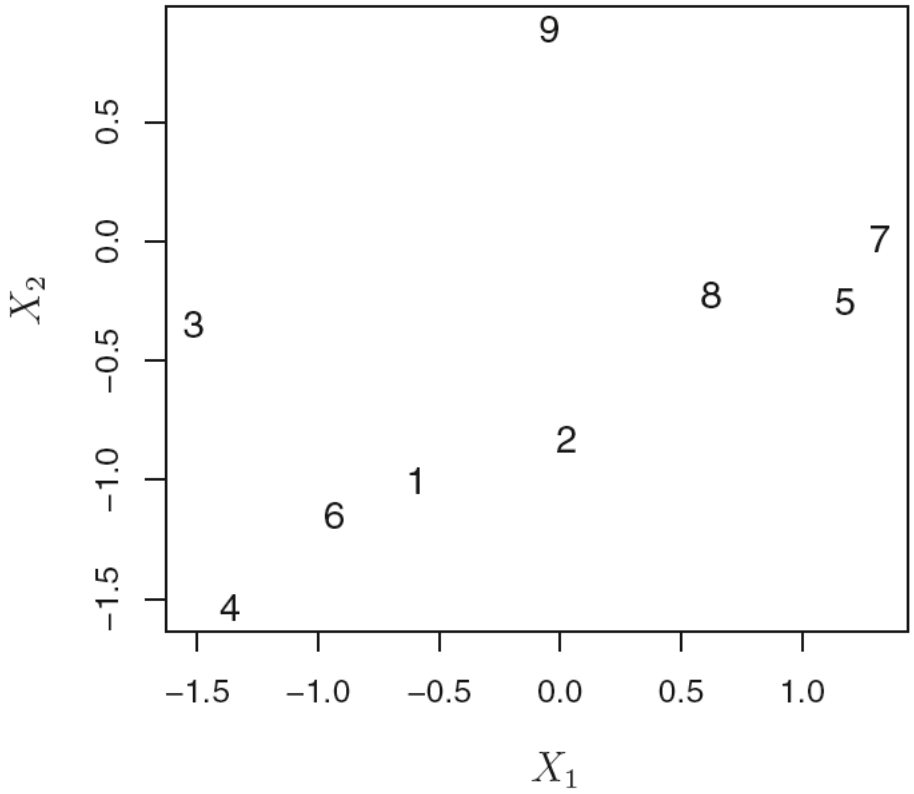


Figure10.10, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Algorithm 10.2 *Hierarchical Clustering*

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 $i = n, n-1, \ldots, 2$:

   (a) 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.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

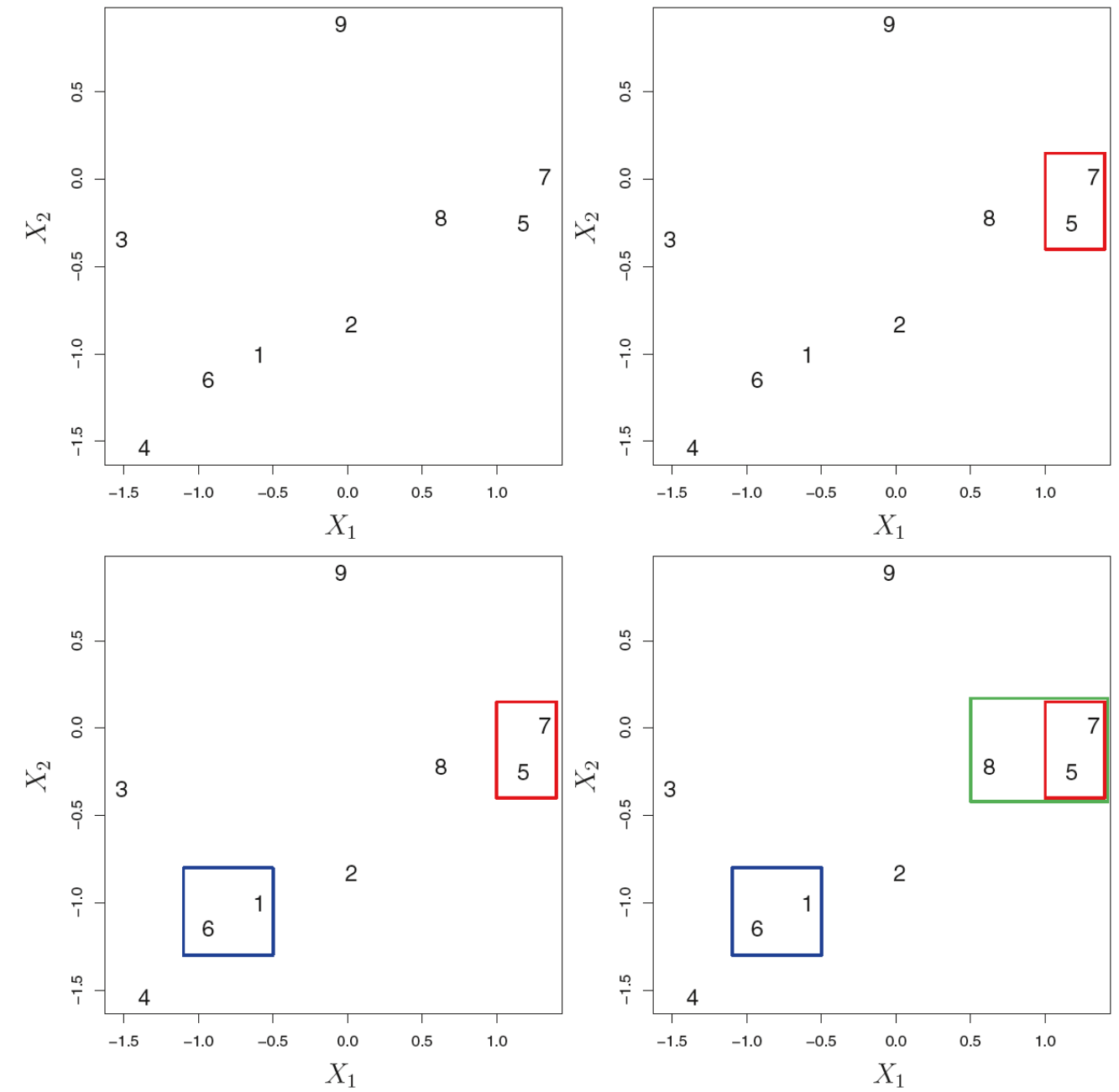


Figure10.11, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Linkage

- dissimilarity between two groups of observations

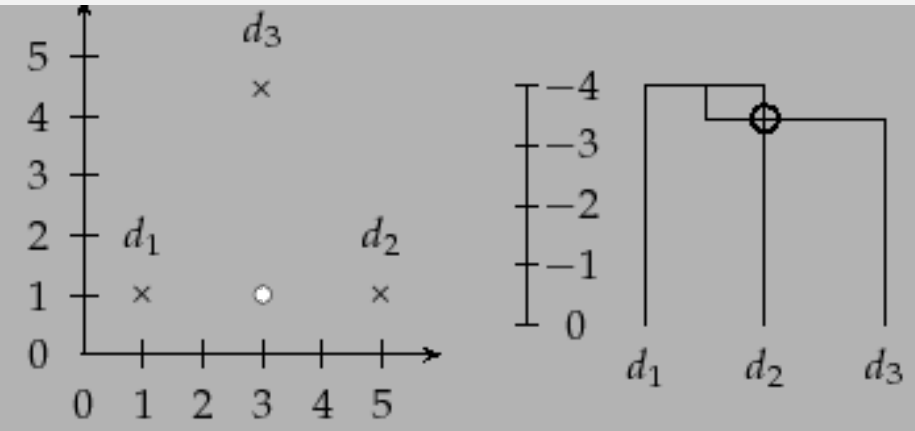
Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Linkage comparison

- *Average*, *complete*, and *single* linkage are most popular among statisticians.
- *Average* and *complete* linkage are generally preferred over *single* linkage, as they tend to yield more balanced dendrograms.

Linkage comparison

- *Centroid* linkage is often used in genomics
  - drawback : an inversion can occur, whereby inversion two clusters are fused at a height below either of the individual clusters in the dendrogram



► **Figure 17.9** Centroid clustering is not monotonic. The documents d_1 at $(1+\epsilon, 1)$, d_2 at $(5, 1)$, and d_3 at $(3, 1+2\sqrt{3})$ are almost equidistant, with d_1 and d_2 closer to each other than to d_3. The non-monotonic inversion in the hierarchical clustering of the three points appears as an intersecting merge line in the dendrogram. The intersection is circled.

Credit by https://nlp.stanford.edu/IR-book/html/htmledition/centroid-clustering-1.html

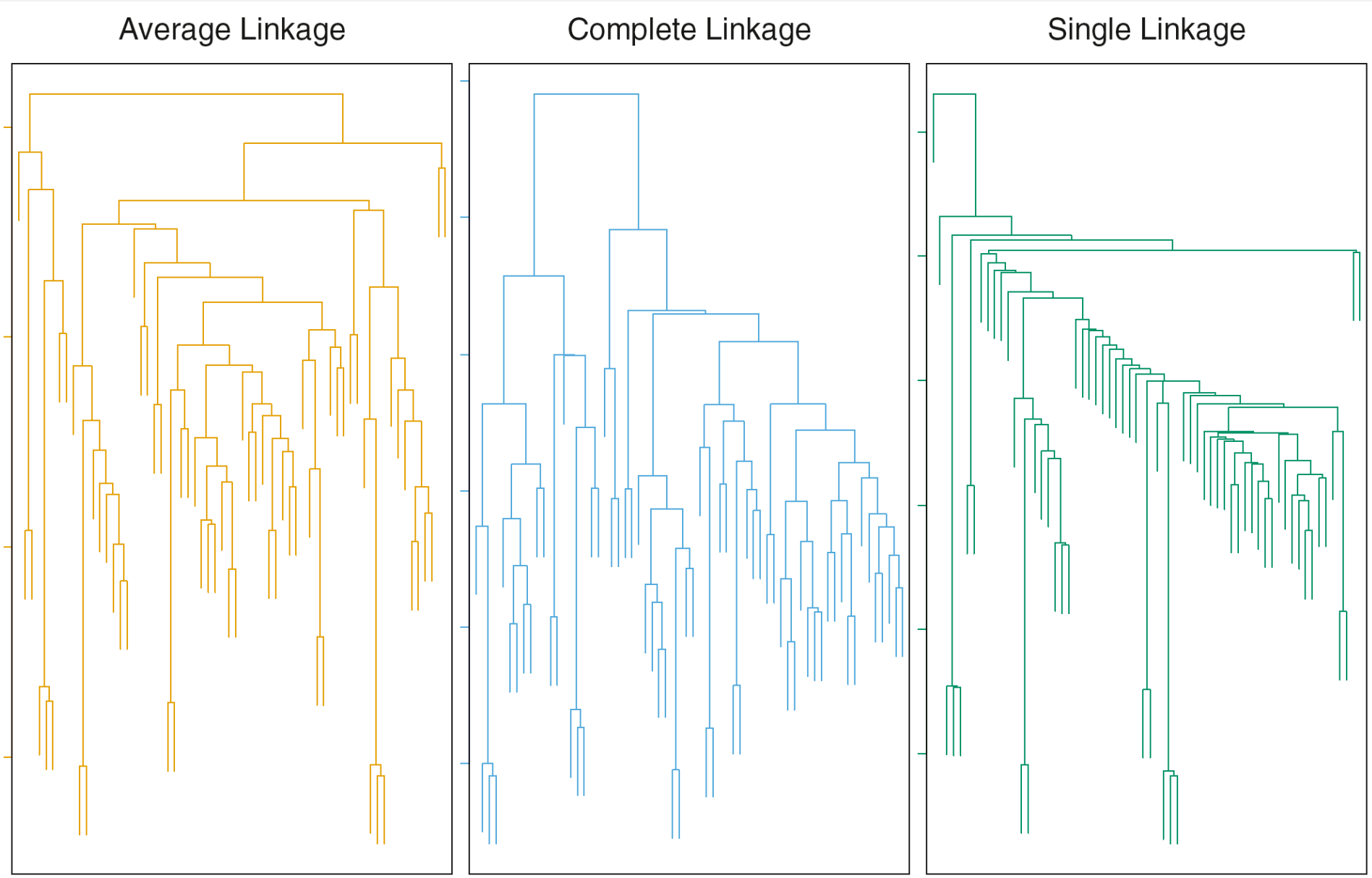


Figure10.12, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Hierarchical clustering

```
d <- dist(pmatrix,
method="euclidean")
pfit <- hclust(d,method="complete")
plot(pfit, labels=protein$Country)
```

Hierarchical clustering

- The dendrogram suggests five clusters = > draw the rectangles on the dendrogram

```
rect.hclust(pfit, k=5)
```

Hierarchical clustering

- Extracting the clusters found by *hclust*

```
groups <- cutree(pfit, k=5)
print_clusters <- function(labels, k) {
   for(i in 1:k) {
      print(paste("cluster", i))
      print(protein[labels==i,c("Country","
      RedMeat","Fish","Fr.Veg")])
   }
}
print_clusters(groups, 5)
```

Visualizing clusters by PCA

```
library(ggplot2)
princ <- prcomp(pmatrix)
nComp <- 4
project <- predict(princ,
newdata=pmatrix)[,1:nComp]
```

Visualizing clusters by PCA

```
project.plus <- cbind(as.data.frame(project),
cluster=as.factor(groups),
country=protein$Country)

ggplot(project.plus, aes(x=PC3, y=PC4)) +
geom_point(aes(shape=cluster)) +
geom_text(aes(label=country),
hjust=0, vjust=1)
```

Complete Linkage with Correlation-Based Distance

```
x=matrix (rnorm (30*3) , ncol =3)
dd=as.dist(1- cor(t(x)))
plot(hclust (dd, method ="complete"),
xlab="", sub ="")
```

k-means

Unsupervised learning

Clustering analysis

K-means clustering

- a simple and elegant approach for partitioning a data set into *K* distinct, non-overlapping clusters

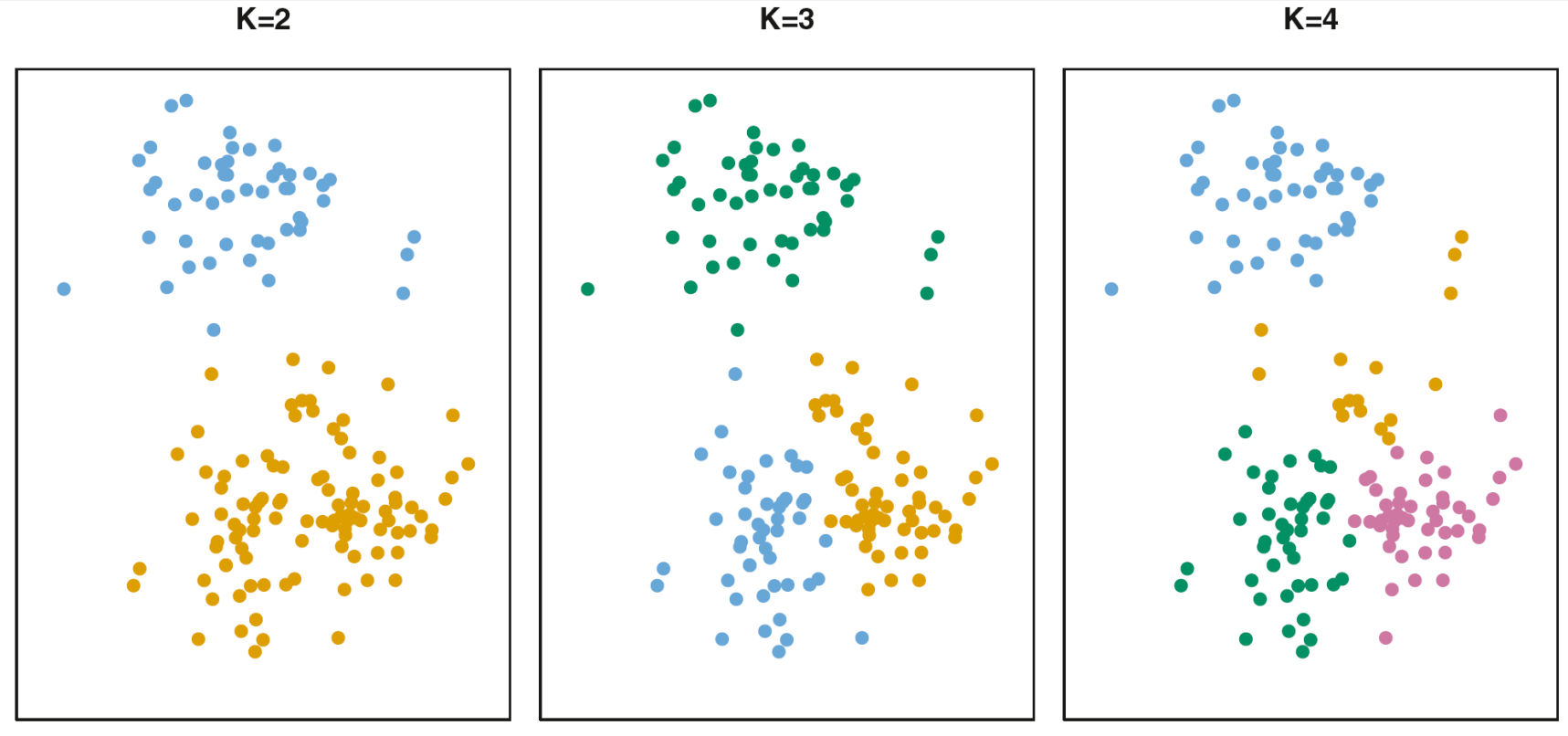


Figure10.5, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

K-means clustering

- Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster.
- $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$.
- $C_i \cap C_j = \varnothing$ for all $i != j$.

Good clustering

- the within-cluster variation is as small as possible

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

Algorithm 10.1 *K-Means Clustering*

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).

The progress of the *K*-means algorithm

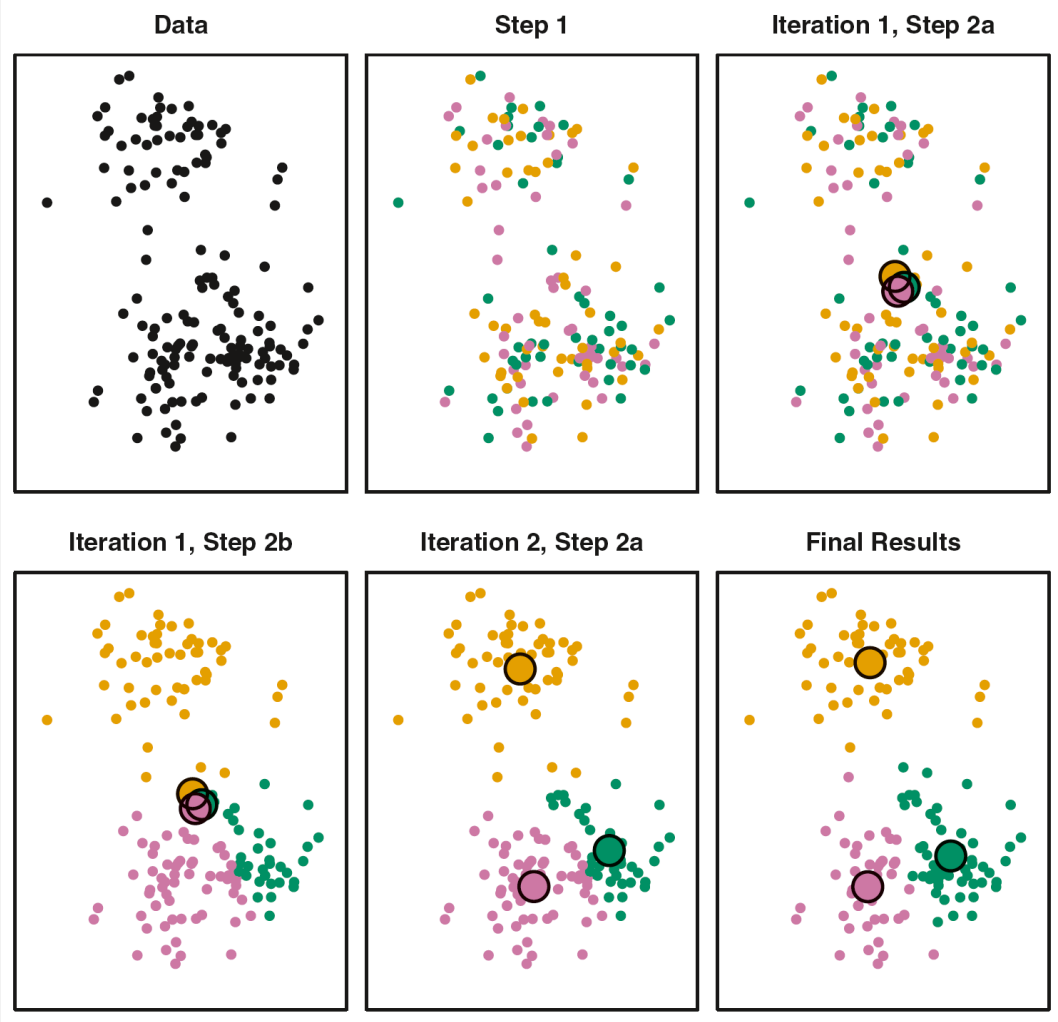


Figure10.6, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

K-means clustering performed 6 times

Figure10.7, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

The *k*-means procedure

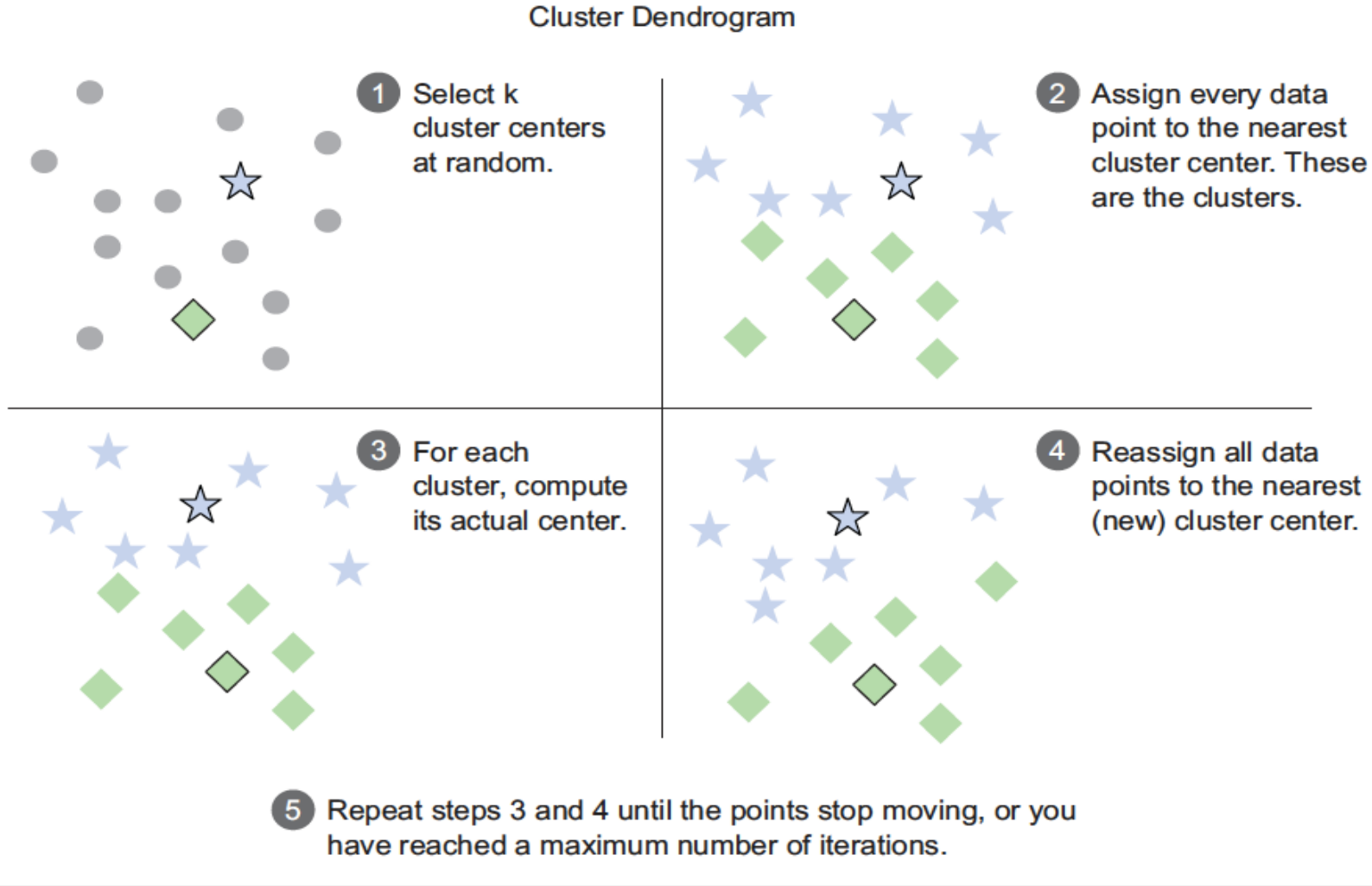


Figure8.6, *Practical Data Science with R* by Nina Zumel, John Mount

Running *k*-means with $k = 5$

```
library(fpc)
kbest.p<-5
pclusters <- kmeans(pmatrix, kbest.p,
nstart=100, iter.max=100)
pclusters
pclusters$centers
pclusters$size
groups <- pclusters$cluster
print_clusters(groups, kbest.p)
```

nstart

- `pclusters <- kmeans(pmatrix, kbest.p, nstart=100, iter.max=100)`
- `pclusters$tot.withinss`
- `pclusters <- kmeans(pmatrix, kbest.p, nstart=1, iter.max=100)`
- `pclusters$tot.withinss`

nstart > 1

- *K*-means clustering will be performed using multiple random assignments in Step 1 of Algorithm 10.1
- the kmeans() function will report only the best results.
- Recommend: 20 or 50

Algorithm 10.1 *K-Means Clustering*

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).

set.seed()

```
set.seed(3)
pclusters <- kmeans(pmatrix, kbest.p, nstart=5,
iter.max=100)
pclusters$tot.withinss
```

the initial cluster assignments in Step 1 can be replicated

=> the *K*-means output will be fully reproducible

Algorithm 10.1 *K-Means Clustering*

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:

The *k*-means algorithm

- the data is all numeric and the distance metric is squared Euclidean (though you could in theory run it with other distance metrics)
- Plus side
  - easy to implement
  - can be faster than hierarchical clustering on large datasets
  - works best on data that looks like a mixture of Gaussians

The *k*-means algorithm

- Negative side
  - must pick *k* in advance
  - fairly unstable
    - the final clusters depend on the initial cluster centers.
    - This algorithm isn't guaranteed to have a unique stopping point.

How many clusters should we look for in the data?

Picking the number of clusters

- total Within Sum of Squares (WSS)
  - centroid : the mean value of all the points in the cluster
  - sum of squares for a single cluster : the average squared distance of each point in the cluster from the cluster's centroid.
  - sum of the within sum of squares of all the clusters

Calculating WSS – WSS.R

```
sqr_edist <- function(x, y) {
      sum((x-y)^2)
}
wss.cluster <- function(clustermat) {
      c0 <- apply(clustermat, 2, FUN=mean)
      sum(apply(clustermat, 1, FUN=function(row){sqr_edist(row,c0)}))
}
wss.total <- function(dmatrix, labels) {
      wsstot <- 0
      k <- length(unique(labels))
      for(i in 1:k)
            wsstot <- wsstot + wss.cluster(subset(dmatrix, labels==i))
      wsstot
}
```

the within-cluster variance, *W*

- $\frac{\text{WSS}(k)}{(n-k)}$
  - where n is the number of points in the dataset
- will decrease as k increases
- the rate of decrease should slow down past the optimal k .

Total Sum of Squares (TSS)

- the squared distance of all the data points from the dataset's centroid

```
totss <- function(dmatrix) {
    grandmean <- apply(dmatrix, 2, FUN=mean)
    sum(apply(dmatrix, 1,
    FUN=function(row){sqr_edist(row, grandmean)}))
}
```

Between Sum of Squares BSS(k)

- Between Sum of Squares BSS(*k*) of the clustering
  - BSS(*k*) = TSS - WSS(*k*)
- The between-cluster variance, *B*
  - BSS(*k*)/(*k*-1)
  - will increase as *k*
  - the rate of increase should slow down past the optimal *k*

Calinski-Harabasz index

- B / W : should be maximized at the optimal *k*
  - B = BSS(*k*)/(*k*-1)
  - W = WSS(*k*)/(*n*-*k*)
  - the between-cluster variance (which is essentially the variance of all the cluster centroids from the dataset's grand centroid) / the total within-cluster variance (the average WSS of the clusters in the clustering)

The Calinski-Harabasz index - CH.R

```
ch_criterion <- function(dmatrix, kmax, method="kmeans") {
      if(!(method %in% c("kmeans", "hclust"))) {
            stop("method must be one of c('kmeans', 'hclust')")
      }
      npts <- dim(dmatrix)[1] # number of rows.
      totss <- totss(dmatrix)
      wss <- numeric(kmax)
      crit <- numeric(kmax)
      wss[1] <- (npts-1)*sum(apply(dmatrix, 2, var))
      for(k in 2:kmax) {
            if(method=="kmeans") {
                  clustering<-kmeans(dmatrix, k, nstart=10, iter.max=100)
                  wss[k] <- clustering$tot.withinss
            }else { # hclust
                  d <- dist(dmatrix, method="euclidean")
                  pfit <- hclust(d, method="ward")
                  labels <- cutree(pfit, k=k)
                  wss[k] <- wss.total(dmatrix, labels)
            }
      }
      bss <- totss - wss
      crit.num <- bss/(0:(kmax-1))
      crit.denom <- wss/(npts - 1:kmax)
      list(crit = crit.num/crit.denom, wss = wss, totss = totss)
}
```

Evaluating clusterings with different numbers of clusters

```
library(reshape2)

source("../CH.R")

source("../WSS.R")

clustcrit <- ch_criterion(pmatrix, 10, method="hclust")

critframe <- data.frame(k=1:10, ch=scale(clustcrit$crit),
wss=scale(clustcrit$wss))

critframe <- melt(critframe, id.vars=c("k"),
variable.name="measure", value.name="score")

ggplot(critframe, aes(x=k, y=score, color=measure)) +

geom_point(aes(shape=measure)) +
geom_line(aes(linetype=measure)) +

scale_x_continuous(breaks=1:10, labels=1:10)
```

the KMEANSRUNS function for picking *K*

```
clustering.ch <- kmeansruns(pmatrix, krange=1:10,
criterion="ch")

clustering.ch$bestk

clustering.ch$crit

critframe <- data.frame(k=1:10, ch=scale(clustering.ch$crit),

asw=scale(clustering.asw$crit))

critframe <- melt(critframe, id.vars=c("k"),
variable.name="measure", value.name="score")

ggplot(critframe, aes(x=k, y=score, color=measure))
+geom_point(aes(shape=measure)) +
geom_line(aes(linetype=measure)) +
scale_x_continuous(breaks=1:10, labels=1:10)
```

Assigning new points to clusters

assign.R

```
assign_cluster <- function(newpt, centers,
xcenter=0, xscale=1) {
    xpt <- (newpt - xcenter)/xscale
    dists <- apply(centers, 1,
    FUN=function(c0){sqr_edist(c0, xpt)})
    which.min(dists)
}
```

Whether a given cluster is *real*?

Jaccard coefficient

- whether a cluster represents true structure is to see if the cluster holds up under plausible variations in the dataset?
- *clusterboot* (fpc package)
  - use bootstrap resampling to evaluate how stable a given cluster is
  - Jaccard coefficient
    - The Jaccard similarity between two sets A and B
    - the number of elements in the intersection of A and B / the number of elements in the union of A and B.
    - Jaccard $= \frac{|A \cap B|}{|A \cup B|}$

Basic general strategy

1. Cluster the data as usual
2. Bootstrap
   - Draw a new dataset (of the same size as the original) by resampling the original dataset with replacement
     - some of the data points may show up more than once, and others not at all
   - Cluster the new dataset
   - For every cluster in the original clustering, find the most similar cluster in the new clustering (the one that gives the maximum Jaccard coefficient) and record that value
     - If this maximum Jaccard coefficient is less than 0.5, the original cluster is considered to be dissolved.
3. Repeat Bootstrap several times

cluster stability by Jaccard coefficient

- The cluster stability of each cluster in the original clustering
  - the mean value of its Jaccard coefficient over all the bootstrap iterations.
- stability value
  - 0.6 <= : unstable.
  - 0.6 ~ 0.75 : measure a pattern in the data, but there isn't high certainty about which points should be clustered together.
  - 0.85 >= : highly stable (they're likely to be real clusters).

clusterboot with heretical clustering

```
library("fpc")
cboot.hclust <-
clusterboot(pmatrix,clustermethod=hclustCBI,method="ward.D",
k=kbest.p)
summary(cboot.hclust$result)
groups<-cboot.hclust$result$partition
print_clusters(groups, kbest.p)
cboot.hclust$bootmean
cboot.hclust$bootbrd
```

clusterboot with *k*-means

```
cboot<-clusterboot(pmatrix, clustermethod=kmeansCBI,
runs=100,iter.max=100, krange=kbest.p, seed=15555)

groups <- cboot$result$partition

print_clusters(cboot$result$partition, kbest.p)

cboot$bootmean

cboot$bootbrd
```

Clustering takeaway

Practical issue

- What dissimilarity measure should be used?
- What type of linkage should be used?
- Where should we cut the dendrogramin order to obtain clusters?

Hierarchical vs *K*-means

- hierarchical : clusters are necessarily nested within the clusters obtained by cutting the dendrogram at any greater height
- hierarchical clustering can sometimes yield worse (i.e. less accurate) results than *K*-means clustering for a given number of clusters

Hierarchical vs *K*-means

- our observations correspond to a group of people
  - with a 50–50 split of males and females
  - evenly split among Americans, Japanese, and French
  - Consequently, this situation could not be well-represented by hierarchical clustering.

Clustering takeaways

- The goal of clustering is to discover or draw out similarities among subsets of your data.
- In a good clustering, points in the same cluster should be more similar (nearer) to each other than they are to points in other clusters.

Clustering takeaways

- When clustering, the units that each variable is measured in matter. Different units cause different distances and potentially different clusterings.
- Ideally, you want a unit change in each coordinate to represent the same degree of change. One way to approximate this is to transform all the columns to have a mean value of 0 and a standard deviation of 1.0, for example by using the function scale() .

Clustering takeaways

- Clustering is often used for data exploration or as a precursor to supervised learning methods.
- Like visualization, it's more iterative and interactive, and less automated than supervised methods.

Clustering takeaways

- Different clustering algorithms will give different results. You should consider different approaches, with different numbers of clusters.
- There are many heuristics for estimating the best number of clusters. Again, you should consider the results from different heuristics and explore various numbers of clusters.

Small Decisions with Big Consequences

- Should the observations or features first be standardized in some way?
  - the variables should be centered to have mean zero and scaled to have standard deviation one

Association rule

Unsupervised learning

Association rules

- used to find objects or attributes that frequently occur together
  - products that are often bought together during a shopping session
- transaction : the unit of “togetherness” when mining association rules
- as items in an itemset : the objects that comprise a transaction are referred

Table 8.1 A database of library transactions

Transaction ID	Books checked out
1	*The Hobbit*, *The Princess Bride*
2	*The Princess Bride*, *The Last Unicorn*
3	*The Hobbit*
4	*The Neverending Story*
5	*The Last Unicorn*
6	*The Hobbit*, *The Princess Bride*, *The Fellowship of the Ring*
7	*The Hobbit*, *The Fellowship of the Ring*, *The Two Towers*, *The Return of the King*
8	*The Fellowship of the Ring*, *The Two Towers*, *The Return of the King*
9	*The Hobbit*, *The Princess Bride*, *The Last Unicorn*
10	*The Last Unicorn*, *The Neverending Story*

Support

- *T*: your database of transactions
- support(*X*)
  - # of transactions that contain X / total # of transactions in *T*

Confidence

- if *X*, then *Y*
  - every time you see the item set *X* in a transaction, you expect to also see *Y*
- conf(X=>Y)
  - support(union(X,Y))/support(X)

Table 8.1 A database of library transactions

Transaction ID	Books checked out
1	*The Hobbit*, *The Princess Bride*
2	*The Princess Bride*, *The Last Unicorn*
3	*The Hobbit*
4	*The Neverending Story*
5	*The Last Unicorn*
6	*The Hobbit*, *The Princess Bride*, *The Fellowship of the Ring*
7	*The Hobbit*, *The Fellowship of the Ring*, *The Two Towers*, *The Return of the King*
8	*The Fellowship of the Ring*, *The Two Towers*, *The Return of the King*
9	*The Hobbit*, *The Princess Bride*, *The Last Unicorn*
10	*The Last Unicorn*, *The Neverending Story*

support of the itemset { The Hobbit, The Princess Bride} = 3/10

confidence : “People who check out The Hobbit also check out The Princess Bride” = 3/5

confidence : “People who check out The Princess Bride also check out The Hobbit ” = 3/4

The goal in association rule mining

- find all the interesting rules in the database with
  - at least a given minimum support: support(X) > 10%
  - a minimum given confidence: conf(X=>Y) > 60%

Book data

- collected in 2004 from the book community Book-Crossing5 for research conducted at the Institut für Informatik, University of Freiburg.
- books/book.R
- Reading in the book data

```
library(arules)
bookbaskets <- read.transactions("bookdata.tsv.gz",
format="single",
sep="\t",
cols=c("userid", "title"),
rm.duplicates=T,
Header=T)
```

Examining the transaction data

```
class(bookbaskets)
bookbaskets
dim(bookbaskets)
colnames(bookbaskets)[1:5]
rownames(bookbaskets)[1:5]
#the distribution of transaction sizes
basketSizes <- size(bookbaskets)
summary(basketSizes)
```

Examining data

```
quantile(basketSizes, probs=seq(0,1,0.1))
library(ggplot2)
ggplot(data.frame(count=basketSizes))+
geom_density(aes(x=count), binwidth=1)+
scale_x_log10()
```

Frequent books

```
bookFreq <- itemFrequency(bookbaskets)
sum(bookFreq)
bookCount <-
(bookFreq/sum(bookFreq))*sum(basketSizes)
summary(bookCount)
orderedBooks <- sort(bookCount, decreasing=T)
orderedBooks[1:10]
orderedBooks[1]/dim(bookbaskets)[1]
```

Finding the association rules

- Preprocessing

```
bookbaskets_use <- bookbaskets[basketSizes > 1]
dim(bookbaskets_use)
```

Finding the association rules

- Decide support & confidence
  - supported by at least 100 people :

```
100/dim(bookbaskets_use)[1]
```

  - confidence : 75%

```
rules <- apriori(bookbaskets_use, parameter
=list(support = 0.002, confidence=0.75))
summary(rules)
```

Scoring rules

```
measures <- interestMeasure(rules,
measure=c("coverage", "fishersExactTest"),
transactions=bookbaskets_use)
```

- coverage: the support of the left side of the rule (*X*)
- fishersExactTest: a significance test for whether an observed pattern is real, or chance (the same thing lift measures; Fisher's test is more formal)
  - Lift = support(union(X, Y))/(support(X)*support(Y))

The five most confident rules discovered in the data

```
inspect(head((sort(rules, by="confidence")), n=5))
```

Finding rules with restrictions

```
brules <- apriori(bookbaskets_use,
parameter =list(support = 0.001, confidence=0.6),
appearance=list(rhs=c("The Lovely Bones: A Novel"),
default="lhs"))
summary(brules)
inspect(head(lhs(brules), n=5))
```

Finding rules with restrictions

- Inspecting rules

```
brulesConf <- sort(brules, by="confidence")
inspect(head(lhs(brulesConf), n=5))
```

- Inspecting rules with restrictions

```
brulesSub <- subset(brules, subset=!(lhs %in% "Lucky
: A Memoir"))
brulesConf <- sort(brulesSub, by="confidence")
inspect(head(lhs(brulesConf), n=5))
```

Association rule takeaways

- The goal of association rule mining is to find relationships in the data: items or attributes that tend to occur together.
- A good rule "if *X*, then *Y*" should occur more often than you'd expect to observe by chance.
- You can use *lift* or *Fisher's exact test* to check if this is true.

Association rule takeaways

- When a large number of different possible items can be in a basket (in our example, thousands of different books), most events will be rare (have low support).
- Association rule mining is often interactive, as there can be many rules to sort and sift through

Takeaways

- Unsupervised methods find structure in the data, often as a prelude to predictive modeling.
- The goal of clustering is to discover or draw out similarities among subsets of your data.

Takeaways

- When clustering, you'll find that scaling is important.
- The goal of association rule mining is to find relationships in the data: items or attributes that tend to occur together.
- In association rule mining, most events will be rare, so support and confidence levels must often be set low.

AITC
教育部人工智慧技術及應用人才培育計畫
Artificial Intelligence Talent Cultivation Program