Dan A. Simovici

Algorithms in Data Analysis Using R version 5

May 14, 2014

Contents

1

Vectors and Matrices in R

1.1 Vectors in R

Vectors are data structures that accommodate sequences of elements that have the same type. A *scalar* in **R** is a vector of length 1.

A scalar is created using an assignment as we show next.

```
x <- 6.9
```

Its length obtained by `length(x)` is 0.

A vector of length n can be defined using the concatenation function `c`. To create the vector $(10, 20, 25)$ we write

```
y <- c(10,20,25)
```

The length of `y` is 3.

Individual components of a vector `v` can be accessed using the notation $v[i]$. For example,

```
> y[2]
```

returns 20.

Vector components can be named. Consider the vector `perfsq`. To assign names to its components we write:

```
perfsq <- c(1,4,9,16,25,36,49,64,81)
> names(perfsq) <- c("one","two","three","four","five","six","seven",
+ "eight","nine")
```

When the value of `perfsq` is inspected we get both the components of the vector and their names:

```
> perfsq
  one   two three  four  five   six seven eight  nine
    1     4     9    16    25    36    49    64    81
```

Portion of a vector defined by subsets of its index values can be extracted by

```
> perfs[2:5]
```

returning

```
[1] 4 9 16 25
```

Random samples can be constructed using the function `sample`.

Example 1. Let `x` be a vector of 10 integer created by

```
x <- 1:10
```

To sample five of its components we write

```
y <- sample(x,5)
```

which may return

```
> y
[1]  6  9 10  7  8
```

If the second argument is omitted, we obtain a random permutation of `x` as in

```
> z <- sample(x)
> z
 [1]  8  4 10  6  5  9  3  7  1  2
```

Samples with replacement can be obtained by using the parameter `replace = TRUE`. For example, we have

```
 w <- sample(x,replace=TRUE)
> w
 [1]  8 10 10 10  3  9  1  8  5  7
```

Finally, to produce a quasi-random sequence of 20 binary digits we write

```
> sample(c(0,1),20,replace=TRUE)
 [1] 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 1 1 0
```

Let us consider other useful examples of manipulating vectors. Note that the concatenation operation c is associative.

Example 2. If we define the vector `ps` as

```
ps <- c(1,4,9,16,25,36,49,64,81)
```

and then write

```
ps <- c(ps,c(8,27,64,125))
```

the result is equivalent to writing

```
ps <- c(1,4,9,16,25,36,49,64,81,8,27,64,125)
```

A vector can be involved in a condition. This condition will be tested for each of its component and a vector of Boolean values will be generated, as it is shown below:

```
> v<- c(1,5,2,4,9,6)
> v <= 6
[1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

The condition can be used to extract the components of the vector that satisfy a condition as in

```
> v[v<6]
[1] 1 5 2 4
```

Example 3. Let us index the components of `ps` with numbers between 1 and 13:

```
> names(ps) <- 1:13
> ps
  1   2   3   4   5   6   7   8   9  10  11  12  13
  1   4   9  16  25  36  49  64  81   8  27  64 125
```

To extract the components of `ps` corresponding to the fourth to the seventh component we can write

```
> idx <- 4:7
> ps[idx]
 4  5  6  7
16 25 36 49
```

To eliminate all components of `ps` between the 4th and the 7th we can use negative indexes:

```
> ps[-idx]
  1   2   3   8   9  10  11  12  13
  1   4   9  64  81   8  27  64 125
```

Similarly, to obtain components of `ps` outside the range from 2 to 7 of the index we can use:

```
> ps[-2:-7]
  1   8   9  10  11  12  13
  1  64  81   8  27  64 125
```

A vector can be sorted using the function `sort`. When applied to `ps` we obtain

```
> pt <- sort(ps)
> pt
  1   2  10   3   4   5  11   6   7   8  12   9  13
  1   4   8   9  16  25  27  36  49  64  64  81 125
```

This allows us to identify the indexes of `ps` that correspond to the components that are less than 40 by writing

```
pt[pt < 40]
```

which yields:

```
 pt[pt < 40]
 1  2 10  3  4  5 11  6
 1  4  8  9 16 25 27 36
```

Finally, to extract the indices of the original vector that correspond to components less than 40 we write:

```
ll <- as.integer(names(pt[pt<40]))
```

which gives the desired answer:

```
> ll
[1]  1  2 10  3  4  5 11  6
```

1.2 Lists

Lists are structures similar to vectors. The main difference is that there is no homogeneity requirements for lists. In other words, list elements may be numbers, strings, or logical values.

Unlike vectors (which can be created using the `c()` functions, lists are created using the function `list`. Elements can be named, which allows a dual access for list components: either by name or by their numbered position. For instance, a list created as

```
> student <- list(name = "John Doe", grad = TRUE, gpa = 3.7)
```

allows us to access its first component either by

```
> student$name
[1] "John Doe"
```

or by

```
> student[1]
$name
[1] "John Doe"
```

1.3 Arrays

Arrays are structures that accomodate multidimensional collections of data. Array components are accessed using the square bracket notation.

To create an array that has three dimension containing $4 \times 2 \times 5$ elements (40 elements in total), we write

```
> a <- array(1:40,c(4,2,5))
```

The content of this array is shown below. Note that for each of the five values of the last index we have a 4×2 array, as shown below.

```
> a
, , 1

     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8

, , 2

     [,1] [,2]
[1,]    9   13
[2,]   10   14
[3,]   11   15
[4,]   12   16

, , 3

     [,1] [,2]
[1,]   17   21
[2,]   18   22
[3,]   19   23
[4,]   20   24

, , 4

     [,1] [,2]
[1,]   25   29
[2,]   26   30
[3,]   27   31
[4,]   28   32

, , 5

     [,1] [,2]
[1,]   33   37
```

```
[2,]   34   38
[3,]   35   39
[4,]   36   40
```

To extract the subarray that corresponds to the value 1 of the first index and the value 2 of the last we write

```
> a[1, ,2]
```

which results in

```
[1]  9 13
```

1.4 Matrices

Matrices are bidimensional arrays. Data is entered in matrices columnwise. If we write

```
> m <- matrix(c(1,2,3,4,5,6),nrow=3)
```

we obtain the 3×2 matrix:

```
> m
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Similarly,

```
p <- matrix(c(2,4,6,5,7,9),ncol=3)
```

creates the matrix 2×3 matrix:

```
> p
     [,1] [,2] [,3]
[1,]    2    6    7
[2,]    4    5    9
```

To compute the product of these matrices we write

```
q <- m %*% p
r <- p %*% m
```

which yield the matrices

```
> q
     [,1] [,2] [,3]
[1,]   18   26   43
[2,]   24   37   59
[3,]   30   48   75

> r
     [,1] [,2]
[1,]   35   80
[2,]   41   95
```

Rows and columns can be added to a matrix using the function `rbind` and `cbind`, respectively.

Example 4. We can generate a two-column matrix that contains the coordinates of 50 points in $\mathbb{R}^2$ centered around $(0,0)$, distributed according to a normal distribution $N(0, 0.3)$ by writing

```
matrix(rnorm(100,sd=0.3), ncol = 2)
```

Another 50 points centered around $(1,1)$ can be obtained as

```
matrix(rnorm(100,mean =1, sd=0.3),ncol = 2)
```

To put together these matrices we can write

```
 x <- rbind(matrix(rnorm(100,sd=0.3),ncol=2),
+ matrix(rnorm(100,mean=1,sd=0.3),ncol=2))
```

and thus, obtain a matrix x with 100 rows and two columns.

To compute the determinant of a square matrix defined by

```
s <- matrix(c(1,0,-1,2,3,4,1,8,9),nrow=3)
```

we write

```
> det(s)
[1] -18
```

The inverse of of a matrix can be computed using the function `ginv` of the library `MASS`.

Example 5.
```
> library(MASS)
> ginv(s)
           [,1]       [,2]       [,3]
[1,]  0.2777778  0.7777778 -0.7222222
[2,]  0.4444444 -0.5555556  0.4444444
[3,] -0.1666667  0.3333333 -0.1666667
>
```

The transpose of a matrix is computed using the function t.

```
X <- matrix(c(1,2,3,4,5,6),nrow=2)
```

Example 6.
```
> X <- matrix(c(1,2,3,4,5,6),nrow=2)
> X
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> ncol(X)
```

```
[1] 3
> Y <- t(X)
> Y
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

The distance between rows of a matrix can be computed by the function `dist` which produces a triangular object as in

```
> m <- matrix(c(1,2,3,4,2,3,4,5,3,4,5,6),nrow=3)
> m
     [,1] [,2] [,3] [,4]
[1,]    1    4    4    4
[2,]    2    2    5    5
[3,]    3    3    3    6
> dist(m)
         1        2
2 2.645751
3 3.162278 2.645751
```

The value returned by `dist` is useful for various functions in the package `class` that compute various types of classifications. However, if a matrix form of these distances is needed we can use the function `as.matrix`:

```
> as.matrix(dist(m))
         1        2        3
1 0.000000 2.645751 3.162278
2 2.645751 0.000000 2.645751
3 3.162278 2.645751 0.000000
```

1.5 Data Frames

The dataframe is a similar to a matrix, so it is a tabular object. However, unlike matrices, its columns may vary in type.

The definition of a data frame begins with the definitions of the vectors that constitute its columns. Then, these columns are assembled into a data frame using the function `data.frame`.

Example 7. A data frame `wh` can be created as follows:

```
> weight <- c(180,150,210,140,170)
> height <- c(1.78,1.64,1.90,1.50,1.89)
> wh <- data.frame(weight,height)
```

This results in the data frame `wh` shown below:

```
> wh
  weight height
1    180   1.78
2    150   1.64
3    210   1.90
4    140   1.50
5    170   1.89
```

An alternative technique for creating a data frame is reading its content from a csv file by using the function `read.csv`.

Example 8. Suppose that we have a `csv` file named `consEU.csv`. By default, **R** assumes that the file contains a header and, in our case, the header consists of

```
code prot fat
```

Thus, upon writing

```
fatprot <- read.csv("C:/Users/Dan/Desktop/cs724-SPRING2014/handouts/consEU.csv")
```

and

```
> fatprot
```

System **R** will return:

```
   code prot fat
1    AL   97  87
2    AT  107 155
3    BY   88  97
4    BE   97 164
5    BA   86  67
6    BG   79 101
7    HR   74  97
8    CY   99 133
9    CZ   95 121
10   DK  108 135
11   EE   88  96
12   FI  105 127
13   FR  117 164
14   GE   77  58
15   DE   99 142
16   GR  117 146
17   HU   90 145
18   IS  128 143
19   IE  115 135
20   IT  113 158
```

```
21   LV   87 116
22   LT  112 105
23   LU  124 164
24   MK   72 102
25   MT  116 110
26   MD   73  59
27   NL  103 135
28   NO  104 144
29   PL  100 113
30   PT  114 137
31   RO  110 107
32   RU   92  87
33   YU   75 116
34   SK   72 108
35   SI  102 131
36   ES  109 152
37   CH   91 152
```

`fatprot` is a new data frame.

The country codes can be obtained using the "$" operator:

```
fatprot$code
```

which returns

```
> fatprot$code
 [1] AL AT BY BE BA BG HR CY CZ DK EE FI FR GE DE GR HU IS IE IT LV LT LU MK MT
[26] MD NL NO PL PT RO RU YU SK SI ES CH
37 Levels: AL AT BA BE BG BY CH CY CZ DE DK EE ES FI FR GE GR HR HU IE ... YU
```

The categorical attribute `code` has 37 values in its domain; these values are the *levels* of the attribute `code`.

If the operator "$" is used in conjunction with a numerical attribute as in

```
fatprot$fat
```

R returns the list of values that occur in the data frame under `fat`:

```
> fatprot$fat
 [1]  87 155  97 164  67 101  97 133 121 135  96 127 164  58 142 146 145 143 135
[20] 158 116 105 164 102 110  59 135 144 113 137 107  87 116 108 131 152 152
```

Projections of a dataframe can be obtained by enclosing a list of attributes between square brackets. For example, we can write:

```
fatprot[c("fat","prot")]
```

This will return

```
   fat prot
1   87   97
2  155  107
3   97   88
4  164   97
5   67   86
6  101   79
7   97   74
8  133   99
9  121   95
10 135  108
11  96   88
12 127  105
13 164  117
14  58   77
15 142   99
16 146  117
17 145   90
18 143  128
19 135  115
20 158  113
21 116   87
22 105  112
23 164  124
24 102   72
25 110  116
26  59   73
27 135  103
28 144  104
29 113  100
30 137  114
31 107  110
32  87   92
33 116   75
34 108   72
35 131  102
36 152  109
37 152   91
```

Equivalently, we could enter

```
fatprot[2:3]
```

To extract the projection of a selected set of rows of the data frame we need to specify two arrays between the square brackets that designate the projection. To return the `code prot` of the first four rows we can write:

```
> fatprot[c(1,2,3,4),c(1,2)]
  code prot
1   AL   97
2   AT  107
```

```
3   BY   88
4   BE   97
```

To inspect the structure of an **R** object one could use the function `str`.

Example 9. The function call `str(fatprot)` returns

```
> str(fatprot)
'data.frame':   37 obs. of  3 variables:
 $ code: Factor w/ 37 levels "AL","AT","BA",..: 1 2 6 4 3 5 18 8 9 11 ...
 $ prot: int  97 107 88 97 86 79 74 99 95 108 ...
 $ fat : int  87 155 97 164 67 101 97 133 121 135 ...
```

1.6 Numeric Computations in R

R provides the usual arithmetic operators, $+, -, *$, and $\hat{}$, standing for addition, subtraction, multiplication and power. In addition $x\%\%y$ stands for "x modulo y".

Trigonometric computations can be performed using

```
sin cos tan asin acos atan
```

The hyperbolic functions

```
sinh cosh tanh asinh acosh atanh
```

are also present. Exponentials and logarithms are computed with

```
exp log log10 logb,
```

where `log` computes natural logarithms, log10 computes base 10 logarithms, and log2 computes base 2 logarithms. The general form `log(x, base)` computes logarithms with base `base`. In addition, `log1p(x)` computes `log(1+x)` accurately for $|x| << 1$.

The functions `gamma` and `lgamma` compute the Euler function $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}\,dt$ and $\ln|\Gamma(a)|$, respectively, when $a \notin \{n \in \mathbb{Z} \mid n \leqslant 0\}$.

The functions `beta` and `lbeta` compute the beta function $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ and the natural logarithm of the beta function, respectively.

The number of combinations of k among of n objects is computed by `choose(n,k)` and its logarithm is calculated with `lchoose(n,k)`. Finally, `factorial(n)` computes $n!$ and its logarithm is produced by `lfactorial(x)`.

Example 10. An approximation of $\sqrt{1+x}$ can be computed using the binomial series $\sum_{k=0}^{\infty}\binom{\frac{1}{2}}{k}x^k$. For $x = 0.25$ we obtain, using the first 6 terms of this series, the value

```
> k <- 0:5
> sum(choose(1/2,k)* 0.25^k)
[1] 1.118038
```

1.7 Functions in R

If the package matlab is obtained and loaded, it is possible to define the function `centmat` that will compute the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}_n'\mathbf{1}_n$ as

```
> centmat <- function(n)
+ { return (eye(n) - (1/n) * ones(n,n))
+ }
```

using the MATLAB functions `eye` and `ones`.

Build-in **R** functions return, in general a list, whose components can be accessed using the "$" notation. For example, the function `eigen` computes the eigenvalues and the eigenvectors of a matrix m, as sown next.

```
 m <- matrix(c(1,2,5,0,2,1,3,1,4),ncol=3)
> m
     [,1] [,2] [,3]
[1,]    1    0    3
[2,]    2    2    1
[3,]    5    1    4
> eigen(m)
$values
[1]  6.934914  1.598564 -1.533479

$vectors
          [,1]        [,2]       [,3]
[1,] 0.4239582  0.17942954  0.7412273
[2,] 0.3417759 -0.98311922 -0.2423938
[3,] 0.8387185  0.03580004 -0.6259611
```

The components of the list are:

- `values`, a vector containing the eigenvalues of m, sorted in decreasing order, according to their absolute values in the asymmetric case when they might be complex (even for real matrices);
- `vectors`, which is either a square matrix whose columns contain the eigenvectors of m; the vectors are normalized to unit length.

To verify the previous computation we write

```
> r <- eigen(m)
> V <- r$vectors
> lam <- r$values
> D = diag(lam)
> V %*% D %*% ginv(V)
```

which results in:

```
     [,1]          [,2] [,3]
[1,]    1 -2.775558e-16    3
[2,]    2  2.000000e+00    1
[3,]    5  1.000000e+00    4
```

Of course, be sure that you load in advance the package `MASS` which contains the function `ginv`.

Singular value decompositions can be achieved using the function `svd`. For a matrix $M \in \mathbb{C}^{n \times p}$ the format is

```
svd(m, nu = min(n, p), nv = min(n, p))
```

Here `nu` is the number of left singular vectors to be computed which must between 0 and n, while `nv` is the number of right singular vectors to be computed; must be between 0 and p.

The list returned by `svd` contains the components:

- `d`: the vector of singular values;
- `u`: a matrix whose columns are the `nu` left singular vectors:
- `v:` a matrix whose columns are the `nv` right singular vectors.

Example 11. For the matrix defined by

```
> m <- matrix(c(1,0,2,1,1,3,4,5,6),ncol=3)
```

the function call

```
svd(m,nu=2,nv=2)
```

returns

```
$d
[1] 9.4974302 1.6090358 0.4580641

$u
r[,1]       [,2]
[1,] -0.4442681 -0.1088962
[2,] -0.5202254 -0.7767400
[3,] -0.7293774  0.6203359

$v
           [,1]       [,2]
[1,] -0.2003724  0.7033874
[2,] -0.3319451  0.6061838
[3,] -0.9217718 -0.3711973
```

Scaling of a data matrix entails centering the data and, if desired, scaling the columns of the matrix. The default format of the function is

```
scale(m, center = TRUE, scale = TRUE)
```

If the arguments `center` and `scale` are omitted these defaults are applied. The argument `center` may be a vector with a number of components equal to the number of columns (features) of the data matrix, or a logical value. If `center` is a vector, then each column has the corresponding component subtracted from it. If `center` is `TRUE` then the column means are subtracted from the corresponding column.

The value of `scale` specifies how the column scaling is performed. If `scale` is a vector as above, then each column of `m` is divided by the corresponding value from scale. If `scale` is `TRUE` then scaling is done by dividing the centered columns of `m` by their standard deviations. If `scale` is `FALSE`, no scaling is done.

Example 12. Consider the scaling performed by

```
> scale(fatprot[,2-3])
```

This returns a standardized matrix where all columns have a mean of zero and a standard deviation of 1.

```
             prot         fat
 [1,] -0.08009906 -1.20414084
 [2,]  0.56417599  1.14440052
 [3,] -0.65994661 -0.85876711
 [4,] -0.08009906  1.45523687
 [5,] -0.78880162 -1.89488829
 [6,] -1.23979415 -0.72061762
 [7,] -1.56193168 -0.85876711
 [8,]  0.04875595  0.38457831
 [9,] -0.20895407 -0.02987016
[10,]  0.62860350  0.45365306
[11,] -0.65994661 -0.89330448
[12,]  0.43532098  0.17735408
[13,]  1.20845104  1.45523687
[14,] -1.36864916 -2.20572465
[15,]  0.04875595  0.69541467
[16,]  1.20845104  0.83356416
[17,] -0.53109160  0.79902679
[18,]  1.91715360  0.72995204
[19,]  1.07959603  0.45365306
[20,]  0.95074102  1.24801264
[21,] -0.72437411 -0.20255702
[22,]  0.88631352 -0.58246813
[23,]  1.65944358  1.45523687
[24,] -1.69078669 -0.68608024
[25,]  1.14402354 -0.40978126
[26,] -1.62635918 -2.17118728
[27,]  0.30646597  0.45365306
[28,]  0.37089348  0.76448942
```

```
[29,]  0.11318345 -0.30616914
[30,]  1.01516853  0.52272781
[31,]  0.75745851 -0.51339338
[32,] -0.40223659 -1.20414084
[33,] -1.49750417 -0.20255702
[34,] -1.69078669 -0.47885601
[35,]  0.24203847  0.31550357
[36,]  0.69303100  1.04078840
[37,] -0.46666409  1.04078840
attr(,"scaled:center")
     prot       fat
 98.24324 121.86486
attr(,"scaled:scale")
    prot      fat
15.52132 28.95414
>
```

The functions `head(l)` and `tail(l)` return, respectively, an initial and a final part of a list `l`.

Using the `sample` function we can construct the function `rpm` that computes a random permutation matrix as

```
rpm <- function(n) {
require(matlab)
A <- mat.or.vec(n,n)
x <- sample(1:n)
for(i in 1:n) A[i,x[i]] <- 1
return(A)
}
```

The function call `mat.or.vec(n,n)` constructs a $n \times n$ matrix whose entries are 0. Then, matrix elements are modified in positions determined by the random permutation `x`. For example, we can write

```
 P <- rpm(5)
> P
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    0    0    0
[2,]    0    0    1    0    0
[3,]    1    0    0    0    0
[4,]    0    0    0    0    1
[5,]    0    0    0    1    0
```

2

Data Sample Matrices

2.1 Introduction

Matrices are natural tools for organizing data sets. Let such a data set consist of a sequence $\mathcal{E}$ of m vectors of $\mathbb{R}^n$, $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m)$. The jth components $(\boldsymbol{u}_i)_j$ of these vectors correspond to the values of a random variable $\mathcal{V}_j$, where $1 \leq j \leq n$. This data series will be represented as a matrix having m rows $\boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_m$ and n columns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$. We refer to matrices obtained in this manner as *sample matrices*. The number m is the *size* of the sample.

In this chapter we present algebraic properties of vectors and matrices associated with a sample matrix: the mean vector and the covariance matrix.

2.2 The Sample Matrix

Each row vector $\boldsymbol{u}'_i$ corresponds to an experiment E_i in the series of experiments $\mathcal{E} = (E_1, \ldots, E_m)$; the experiment E_i consists of measuring the n components of $\boldsymbol{u}'_i = (x_{i1}, \ldots, x_{in})$, as shown below.

$$\begin{array}{c|ccc}
 & \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n \\
\hline
\boldsymbol{u}'_1 & x_{11} & \cdots & x_{1n} \\
\boldsymbol{u}'_2 & x_{21} & \cdots & x_{2n} \\
\vdots & \vdots & \vdots & \vdots \\
\boldsymbol{u}'_m & x_{m1} & \cdots & x_{mn} \\
\hline
\end{array}$$

The column vector

$$\boldsymbol{v}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{mj} \end{pmatrix}$$

represents the measurements of the jth variable $\mathcal{V}_j$ of the experiment, for $1 \leq j \leq n$, as shown below. These variables are usually referred to as *attributes* or *features* of the series $\mathcal{E}$.

Definition 1. *The* mean *of D is the vector $\tilde{D} = \frac{1}{m} D' \mathbf{1}_m \in \mathbb{R}^n$. D is* centered *if $\tilde{D} = \mathbf{0}_n$.*

In particular, if $D = (\boldsymbol{v}) \in \mathbb{R}^{m \times 1}$, then $\tilde{\boldsymbol{v}} = \frac{1}{m} \boldsymbol{v}' \mathbf{1}_m$.

Example 13. The following data matrix records the weights and heigths of five individuals:

weight (lbs)	height (m)
180	1.78
150	1.64
210	1.90
140	1.50
170	1.89

This data matrix D belongs to $\mathbb{R}^{5 \times 2}$ and has two features: weight and height, and five observations:

$$\begin{pmatrix} 180 \\ 1.78 \end{pmatrix}, \begin{pmatrix} 150 \\ 1.64 \end{pmatrix}, \begin{pmatrix} 210 \\ 1.90 \end{pmatrix}, \begin{pmatrix} 140 \\ 1.50 \end{pmatrix}, \begin{pmatrix} 170 \\ 1.89 \end{pmatrix}$$

We have $D \in \mathbb{R}^{5 \times 2}$, so $\tilde{D} \in \mathbb{R}^2$ is given by

$$\begin{aligned} \tilde{D} &= \frac{1}{5} D' \mathbf{1}_5 \\ &= \frac{1}{5} \begin{pmatrix} 180 & 150 & 210 & 140 & 170 \\ 1.78 & 1.64 & 1.90 & 1.50 & 1.89 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 170 \\ 1.74 \end{pmatrix} \end{aligned}$$

Theorem 1. *Let $D \in \mathbb{R}^{m \times n}$ be a data matrix and let*

$$H_m = I_m - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m' = I_m - \frac{1}{m} J_{m,m}.$$

$H_m D$ is a centered data matrix.

Proof: We have

$$\begin{aligned}(\widetilde{H_m D}) &= \frac{1}{m} D' H_m' \mathbf{1}_m = \frac{1}{m} D' \left(I_m - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m' \right)' \mathbf{1}_m \\ &= \frac{1}{m} D' \left(I_m - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m' \right) \mathbf{1}_m = \frac{1}{m} D' \left(\mathbf{1}_m - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m' \mathbf{1}_m \right) = \mathbf{0}_n.\end{aligned}$$

Theorem 2. *The centering matrix* $H_m = I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m'$ *is both symmetric and idempotent; further,* $H_m\mathbf{1}_m = \mathbf{0}_m$.

Example 14. For $D \in \mathbb{R}^{5\times 2}$ the centering matrix is

$$H_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \frac{1}{5} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1\ 1\ 1\ 1\ 1),$$

so

$$H = \begin{pmatrix} 0.8000 & -0.2000 & -0.2000 & -0.2000 & -0.2000 \\ -0.2000 & 0.8000 & -0.2000 & -0.2000 & -0.2000 \\ -0.2000 & -0.2000 & 0.8000 & -0.2000 & -0.2000 \\ -0.2000 & -0.2000 & -0.2000 & 0.8000 & -0.2000 \\ -0.2000 & -0.2000 & -0.2000 & -0.2000 & 0.8000 \end{pmatrix}.$$

Thus, the centered matrix is

$$H_5 A = \begin{pmatrix} 10.0000 & -0.1220 \\ -20.0000 & -0.0620 \\ 40.0000 & 0.1980 \\ -30.0000 & -0.2020 \\ 0 & 0.1880 \end{pmatrix}$$

Let

$$D = \begin{pmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_m' \end{pmatrix} = (\boldsymbol{v}_1\ \cdots\ \boldsymbol{v}_n) \in \mathbb{R}^{m\times n}$$

be a data matrix and let $\boldsymbol{z} \in \mathbb{R}^n$.

Definition 2. *The* inertia of D relative to $\boldsymbol{z}$ *is the number* $I_{\boldsymbol{z}}(D) = \sum_{j=1}^m \| \boldsymbol{u}_j - \boldsymbol{z} \|_2^2$.

Theorem 3. *(Huygens' Inertia Theorem) Let*

$$D = \begin{pmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_m' \end{pmatrix}$$

be a data matrix. We have

$$I_{\boldsymbol{z}}(D) - I_{\tilde{D}}(D) = m \parallel \tilde{D} - \boldsymbol{z} \parallel_2^2,$$

for every $\boldsymbol{z} \in \mathbb{R}^n$*. The minimal value of the inertia* $I_{\boldsymbol{z}}(D)$ *is achieved for* $\boldsymbol{z} = \tilde{D}$*.*

Proof: The inertia of $\boldsymbol{u}$ relative to $\tilde{\boldsymbol{u}}$ is

$$\begin{aligned} I_{\tilde{\boldsymbol{u}}}(\boldsymbol{u}) &= \sum_{j=1}^{m} \parallel \boldsymbol{u}_j - \tilde{\boldsymbol{u}} \parallel_2^2 \\ &= \sum_{j=1}^{m} (\boldsymbol{u}_j - \tilde{\boldsymbol{u}})'(\boldsymbol{u}_j - \tilde{\boldsymbol{u}}) \\ &= \sum_{j=1}^{m} (\boldsymbol{u}_j'\boldsymbol{u}_j - \tilde{\boldsymbol{u}}'\boldsymbol{u}_j - \boldsymbol{u}_j'\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{u}}'\tilde{\boldsymbol{u}}). \end{aligned}$$

Similarly, we have

$$I_{\boldsymbol{z}}(\boldsymbol{u}) = \sum_{j=1}^{m} (\boldsymbol{u}_j'\boldsymbol{u}_j - \boldsymbol{z}'\boldsymbol{u}_j - \boldsymbol{u}_j'\boldsymbol{z} + \boldsymbol{z}'\boldsymbol{z}).$$

This allows us to write

$$\begin{aligned} I_{\boldsymbol{z}}(\boldsymbol{u}) - I_{\tilde{\boldsymbol{u}}}(\boldsymbol{u}) &= \sum_{j=1}^{m} (\tilde{\boldsymbol{u}} - \boldsymbol{z})'\boldsymbol{u}_j + \sum_{j=1}^{m} \boldsymbol{u}_j'(\tilde{\boldsymbol{u}} - \boldsymbol{z}) + \boldsymbol{z}'\boldsymbol{z} - \tilde{\boldsymbol{u}}'\tilde{\boldsymbol{u}} \\ &= (\tilde{\boldsymbol{u}} - \boldsymbol{z})' \sum_{i=1}^{m} \boldsymbol{u}_j + \left(\sum_{j=1}^{m} \boldsymbol{u}_j \right)' (\tilde{\boldsymbol{u}} - \boldsymbol{z}) + m(\boldsymbol{z}'\boldsymbol{z} - \tilde{\boldsymbol{u}}'\tilde{\boldsymbol{u}}) \\ &= m(\tilde{\boldsymbol{u}} - \boldsymbol{z})'\tilde{\boldsymbol{u}} + m\tilde{\boldsymbol{u}}'(\tilde{\boldsymbol{u}} - \boldsymbol{z}) + m(\boldsymbol{z}'\boldsymbol{z} - \tilde{\boldsymbol{u}}'\tilde{\boldsymbol{u}}) \\ &= m \parallel \tilde{\boldsymbol{u}} - \boldsymbol{z} \parallel_2^2, \end{aligned}$$

which is the equality of the theorem.

Corollary 1. *Let* $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in \mathbf{Seq}_m(\mathbb{R}^n)$*. The minimal value of the inertia* $I_{\boldsymbol{z}}(\boldsymbol{u})$ *is achieved for* $\boldsymbol{z} = \tilde{\boldsymbol{u}}$*.*

Definition 3. *The* standard deviation of a vector $\boldsymbol{v} \in \mathbb{R}^m$ *is the number*

$$s_{\boldsymbol{v}} = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (v_i - \tilde{v})^2},$$

where $\tilde{v} = \frac{1}{m} \sum_{i=1}^{m} v_i$ *is the mean of the components of* $\boldsymbol{v}$*.*

The variance *is* $var(\boldsymbol{v}) = s_{\boldsymbol{v}}^2$*.*

The standard deviation of a data matrix $D \in \mathbb{R}^{m\times n}$, *where* $D = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n)$ *is the row*

$$\boldsymbol{s} = (s_{\boldsymbol{v}_1}, \ldots, s_{\boldsymbol{v}_n}) \in \mathbb{R}^n.$$

The standard deviation of a data matrix $D \in \mathbb{R}^{m\times n}$, *where* $D = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n)$ *is the row* $\boldsymbol{s} = (s_{\boldsymbol{v}_1}, \ldots, s_{\boldsymbol{v}_n}) \in \mathbb{R}^n$.

Let $\boldsymbol{u}$ and $\boldsymbol{w}$ be two vectors in $\mathbb{R}^m$, where $m > 1$, having the means $\tilde{u}$ and $\tilde{w}$, and the standard deviations s_u and s_v, respectively.

The *covariance coefficient* of $\boldsymbol{u}$ and $\boldsymbol{w}$ is the number

$$cov(\boldsymbol{u}, \boldsymbol{w}) = \frac{1}{m-1} \sum_{i=1}^{m-1} (u_i - \tilde{u})(w_i - \tilde{w}).$$

Definition 4. *The* correlation coefficient *of* $\boldsymbol{u}$ *and* $\boldsymbol{w}$ *is the number*

$$\rho(\boldsymbol{u}, \boldsymbol{w}) = \frac{cov(\boldsymbol{u}, \boldsymbol{w})}{s_u s_w}.$$

The covariance matrix *is of a data matrix* $D \in \mathbb{R}^{m\times n}$ *is*

$$cov(D) = \frac{1}{m-1} \tilde{D}' \tilde{D} \in \mathbb{R}^{n\times n}.$$

The total variance $\mathsf{tvar}(D)$ *of* D *is*

$$\mathsf{tvar}(D) = \mathsf{trace}(cov(X)).$$

Theorem 4. *For* $\boldsymbol{v} \in \mathbb{R}^m$ *we have*

$$var(\boldsymbol{v}) = \frac{1}{m-1} \left(\| \boldsymbol{v} \|^2 - m\tilde{v}^2 \right).$$

Theorem 5. *Let* $D \in \mathbb{R}^{m\times n}$ *be a data matrix, where*

$$D = \begin{pmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_m' \end{pmatrix} = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n).$$

The mean square distance between column vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ *is equal to twice the sum of row variances,* $\sum_{i=1}^m var(\boldsymbol{u}_i)$.

Proof: The mean square distance between the columns of D is

$$\begin{aligned}
&\frac{2}{n(n-1)}\sum_{i<j}\parallel \boldsymbol{v}_i-\boldsymbol{v}_j\parallel^2\\
&=\frac{2}{n(n-1)}\left(\sum_{j=1}^{n}\parallel \boldsymbol{v}_j\parallel^2-2\sum_{i<j}\boldsymbol{v}_i'\boldsymbol{v}_j\right)\\
&=\frac{2}{n(n-1)}\left((n-1)\parallel D\parallel_F^2+\parallel D\parallel_F^2-\mathbf{1}_n'DD'\mathbf{1}_n\right)\\
&=\frac{2}{n(n-1)}\left(n\parallel D\parallel_F^2-\mathbf{1}_n'DD'\mathbf{1}_n\right).
\end{aligned}$$

Since each vector $\boldsymbol{u}_k$ belongs to $\mathbb{R}^n$, the the sum of row variances is

$$\sum_{k=1}^{m} var(\boldsymbol{u}_k)=\sum_{k=1}^{m}\frac{1}{n-1}\left(\parallel \boldsymbol{u}_k\parallel^2-n\tilde{u}_k^2\right)=\frac{1}{n-1}\parallel D\parallel_F^2-\frac{n}{n-1}\sum_{k=1}^{n}\tilde{u}_k^2.$$

Taking into account that

$$\tilde{u}_k=\frac{1}{n}\mathbf{1}_n'D'\boldsymbol{e}_k=\frac{1}{n}e_k'D\mathbf{1}_n,$$

we have $\tilde{u}_k^2=\frac{1}{n^2}\mathbf{1}_n'D'\boldsymbol{e}_ke_k'D\mathbf{1}_n$, which implies

$$\sum_{k=1}^{n}\tilde{u}_k^2=\frac{1}{n^2}\mathbf{1}_n'D'\left(\sum_{k=1}^{n}\boldsymbol{e}_k\boldsymbol{e}_k'\right)D\mathbf{1}_n=\frac{1}{n^2}\mathbf{1}_n'D'D\mathbf{1}_n,$$

because $\sum_{k=1}^{n}\boldsymbol{e}_k\boldsymbol{e}_k'=I_m$. The desired equality follows immediately.

Theorem 6. *We have $-1\le\rho(\boldsymbol{u},\boldsymbol{w})\le 1$ for any vectors $\boldsymbol{u},\boldsymbol{w}\in\mathbb{R}^m$.*

Proof: By Cauchy-Schwarz Inequality we have

$$\left|\sum_{i=1}^{m}(u_i-u)(w_i-w)\right|\le\sqrt{\sum_{i=1}^{m}(u_i-u)^2}\cdot\sqrt{\sum_{i=1}^{m}(w_i-w)^2},$$

which implies

$$-1\le\rho(\boldsymbol{u},\boldsymbol{w})\le 1.$$

Theorem 7. *The covariance matrix of a data matrix $D\in\mathbb{R}^{m\times n}$ can be written as*

$$cov(D)=\frac{1}{m-1}D'H_mD;$$

if D is centered, then $cov(D)=\frac{1}{m-1}D'D$.

Theorem 8. *Let*

$$D \in \mathbb{R}^{m\times n} = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n)$$

be a centered data matrix and let $R \in \mathbb{R}^{n\times n}$ be an orthogonal matrix.

The matrix $Z = DR \in \mathbb{C}^{m\times n}$ is centered and $cov(DR) = R' cov(D) R$.

Proof: By writing explicitly the rows of the matrix Z,

$$Z = \begin{pmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_m \end{pmatrix},$$

we have $\boldsymbol{z}_i = \boldsymbol{u}_i R$ for $1 \leq i \leq m$ because $Z = XR$.

Note that the mean of Z is

$$\tilde{Z}' = \frac{1}{m}\mathbf{1}'_m Z = \frac{1}{m}\mathbf{1}'_m DR = \tilde{D}' R.$$

Since D is centered, we have $\tilde{D}' = \mathbf{0}'_n$, so Z is centered as well.

The covariance matrix of Z is

$$cov(Z) = \frac{1}{m-1} Z'Z = \frac{1}{m-1} R'D'DR = R' cov(D) R.$$

Since the trace of two similar matrices are equal and $cov(Z)$ is similar to $cov(D)$, the total variance of Z equals the total variance of D, that is,

$$\mathsf{tvar}(Z) = \mathsf{trace}(cov(Z)) = \mathsf{trace}(cov(D)) = \mathsf{tvar}(D).$$

Since the covariance matrix of a centered matrix D, $cov(D) = \frac{1}{m-1} D'D \in \mathbb{R}^{n\times n}$ is symmetric $cov(X)$ is orthonormally diagonalizable, so there exists an orthogonal matrix $R \in \mathbb{R}^{n\times n}$ such that $R' cov(D) R = D$, which corresponds to a matrix $Z = DR$.

Let $cov(Z) = D = \mathsf{diag}(d_1, \ldots, d_n)$. d_p is the variance of the pth variable of the data matrix, and the covariances of the form $cov(Z)_{pq}$ with $p \neq q$ are 0. From a statistical point of view, this means that the components p and q are uncorrelated.

2.3 Variance and Covariance

The **R** functions `var`, `cov` and `cor` compute the variance of of a vector, the covariance of two vectors, or their correlation, respectively.

Example 15. Let $\boldsymbol{x}, \boldsymbol{y}$ be the vectors defined by

```
> x <- c(1,2,3,4,5)
> y <- c(4,5,1,0,6)
```

The function `cov(x,y)` returns -0.25; the function `cor` returns -0.06108472, and the function `var(x)` returns 2.5.

If M and P are matrices having the same number of rows the covariances or correlations between the columns of M and the columns of P are computed by the functions `cov` and `cor`, respectively.

Example 16. Let M and P be the matrices having the same numbers of rows:

```
> M <- matrix(c(1,0,2,3,4,1,0,-1,0,1,3,-2),ncol=3)
> M
     [,1] [,2] [,3]
[1,]    1    4    0
[2,]    0    1    1
[3,]    2    0    3
[4,]    3   -1   -2
```

and

```
> P <- matrix(c(-1,3,5,0,1,2,4,3),nrow=4)
> P
     [,1] [,2]
[1,]   -1    1
[2,]    3    2
[3,]    5    4
[4,]    0    3
```

then the function `cov(M,P)` returns the covariances between the columns of the of the matrices:

```
> cov(M,P)
          [,1]      [,2]
[1,] -0.500000  1.000000
[2,] -2.666667 -2.333333
[3,]  4.833333  1.000000
```

Similarly, `cor` computes the correlations between the columns as in

```
 > cor(M,P)
           [,1]       [,2]
[1,] -0.1406422  0.6000000
[2,] -0.4482655 -0.8366600
[3,]  0.8431515  0.3721042
```

The function call `cov(M)` returns the variances of the columns of M on the diagonal of the resulting matrix, and the covariances of the columns of M on the other elements, as we show below.

```
> M
     [,1] [,2] [,3]
[1,]    1    4    0
[2,]    0    1    1
[3,]    2    0    3
[4,]    3   -1   -2
> cov(M)
          [,1]       [,2]       [,3]
[1,]  1.666667 -1.6666667 -1.0000000
[2,] -1.666667  4.6666667  0.3333333
[3,] -1.000000  0.3333333  4.3333333
> var(M[,1])
[1] 1.666667
> var(M[,2])
[1] 4.666667
> cov(M[,1],M[,3])
[1] -1
```

3

Graphics in R

3.1 Histograms and Scatterplots

To create a histogram that shows the frequencies of countries where certain levels of consumptions of fats occur we write

```
> hist(fatprot$fat,main="Fat Histogram",xlab="Fets")
```

A similar histogram can be produced for the proteins as

```
hist(fatprot$prot,main="Protein  Histogram",xlab="Proteins")
```

The results are shown in Figure 3.1.

Scatterplots are representations of points in $\mathbb{R}^2$. To illustrate the technique used in these representations, consider a 100×2-matrix S whose rows represent 100 points in $\mathbb{R}^2$. To be able to identify these points we will add a third column to the matrix S containing the row numbers whose enties vary between 1 and 100 using

```
> Z <- cbind(S,1:100)
```

The first fifty points (grouped around the point $(2,2)$) will be represented on the graph by solid circles; the next fifty points (grouped arount the point $(3,4)$) will appear as solid squares. The shape of the points in the plot is specified using the parameter `pch`, whose value is determined by the conditional expression

```
pch=ifelse((Z[,3]<=50),19,22)
```

If the third component of Z is between 1 and 50, `pch` will assume the value 19 (corresponding to a solid circle); if the third component is greater than 50, the value of `pch` will be 22 (corresponding to a square). This the plot is achieved by

```
> plot(Z[,1:2],pch=ifelse((Z[,3]<=50),19,22),xlab="Z1",ylab="Z2")
```

and it is shown in Figure 3.2. Parameters `xlab` and `ylab` provide the needed labels for the axes.

Three lines separating the two sets of points can be drawn using the function `abline`. To draw a full line we can write

Fig. 3.1. Histograms showing the distibutions of fat and protein consumption

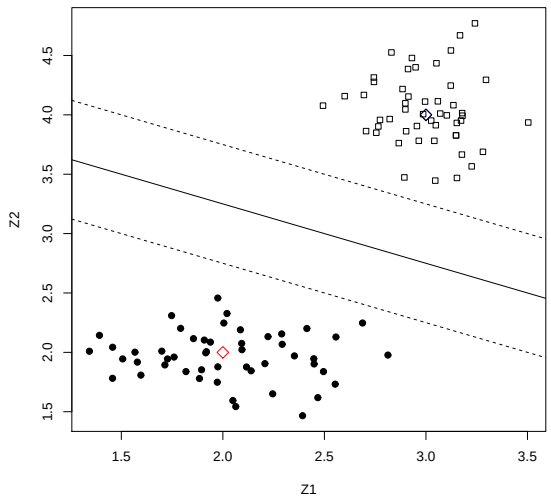

Fig. 3.2. Scatterplot showing a set of points in $\mathbb{R}^2$ and three lines

```
abline(intercept, slope)
```

with an obvious significance. Dotted lines can be drawn adding the parameter `lty`. Possible values for `lty` include one of the character strings "blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash", where "blank" uses invisible lines (i.e., does not draw them), or, equivalently, the numbers from 0 to 6, respectively.

Thus, we placed three lines on the plot by writing

```
> abline(4.25,-0.5)
> abline(4.75,-0.5,lty="dashed")
> abline(3.75,-0.5,lty="dashed")
```

Individual points (shaped as diamonds) were added to mark the centers of the two sets of point using the function `points`:

```
> points(2,2,pch=23,col="red",cex=1.5)
> points(3,4,pch=23,col="black",cex=1.5)
```

The parameter `cex` indicates the amount by which the ploted objects should be scaled relative to the default.

Scatterplots allow the representation of the interdependency between two variables. For example, to represent the interdependency between the protein consumption and the fat consumption we write

```
> plot(prot ~ fat,data=fatprot)
> text(fatprot$fat,fatprot$prot,labels=fatprot$code)
```

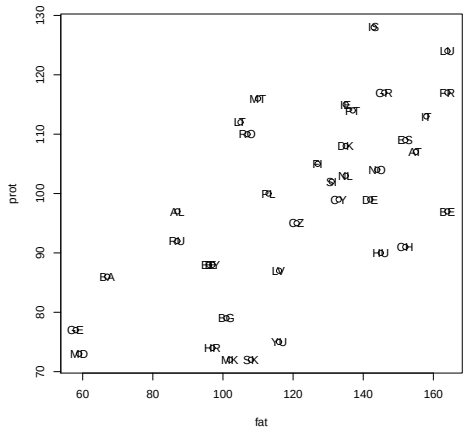

Fig. 3.3. Scatterplot showing the interdependency between protein and fat consumption

The resulting scatter plot is shown in Figure 3.3.

We will discuss one of the most important graphics packages, namely `ggplot2` and will illustrate various graphics constructions with the help of a data set named `diamonds` that comes as a part of this package and contains pricing information for more than 50,000 cut diamonds. After loading this package we can explore this data set by writing:

```
> package ggplot2 was built under R version 3.0.3
> data(diamonds)
> head(diamonds)
  carat       cut color clarity depth table price    x    y    z
1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
```

```
6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
> tail(diamonds)
      carat       cut color clarity depth table price    x    y    z
53935  0.72   Premium     D     SI1  62.7    59  2757 5.69 5.73 3.58
53936  0.72     Ideal     D     SI1  60.8    57  2757 5.75 5.76 3.50
53937  0.72      Good     D     SI1  63.1    55  2757 5.69 5.75 3.61
53938  0.70 Very Good     D     SI1  62.8    60  2757 5.66 5.68 3.56
53939  0.86   Premium     H     SI2  61.0    58  2757 6.15 6.12 3.74
53940  0.75     Ideal     D     SI2  62.2    55  2757 5.83 5.87 3.64
```

To plot the distribution of carats we can write in ggplot2:

```
> ggplot(data = diamonds)+geom_histogram(aes(x=carat))
```

This will result in the plot shown in Figure 3.5

Density plots give the probability of observations falling within an interval of the set of values of a variable. A density plot of the prices can be obtained using

```
> ggplot(data = diamonds)+geom_density(aes(x=carat),fill= "tomato")
```

Here "tomato" is the color used in the plot, shown in Figure 3.6.

The list of avialable colors can be obtained by entering

```
> color()
```

which results in a list of 655 colors.

The parameter `aes` of various geometric layers stands for `aesthetic map`. The "+" sign stands for layer addition.

Scatter plots can be created using the `aes` parameter in the `ggplot` function. For examplr to create a scatter plot showing the dependency of the price on the number of carats we write

```
> scp <- ggplot(diamonds,aes(x=carat,y=price))
```

The object `scp` has no layers yet, so it cannot be examined. To obtain a back and white image we write

```
scp +geom_point()
```

which results in the scatterplot shown in Figure 3.6

To color the points of the scatterplots with a color that depends on the `color` of the diamonds we write

```
scp + geom_point(aes(color=color))
```

This results in a scatterplot; the color of various points is explained in a legend that is automatically generated (see Figure 3.7).

Other packages provide interesting graphical representations. The package `psych` enables us to compute a suggestive visualization of correlations of data matrices. This visualization consists of a collection of scatterplots arranged as a rectangular array. To illustrate we shall use some of the numberical features of the `diamonds` data frame provided by the package `ggplot2`, namely `carat`, `depth`, and `price`. The function `pairs` is a part of the `psych` package. Thus, as a response to

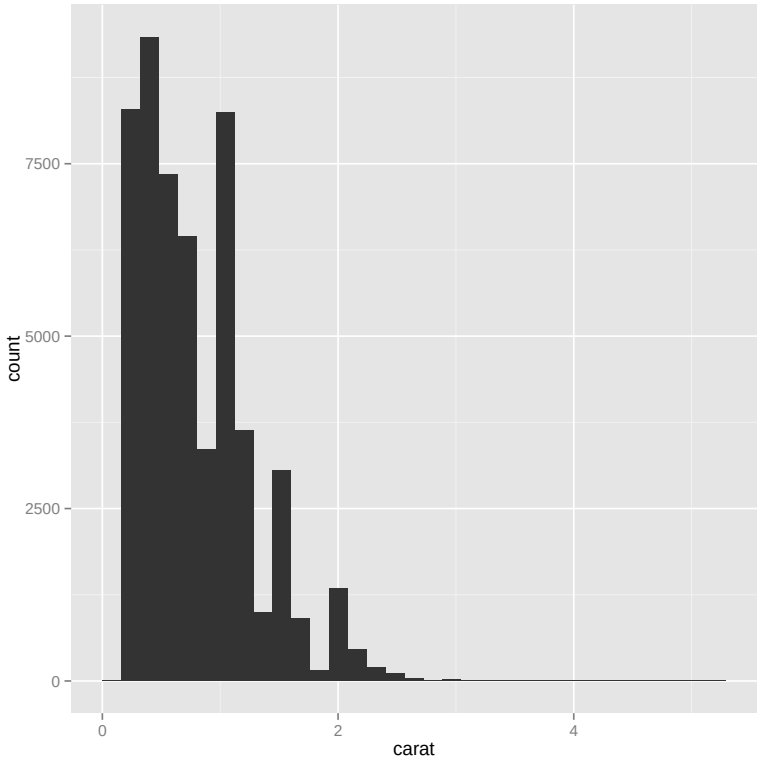

Fig. 3.4. Histogram of carats in diamonds

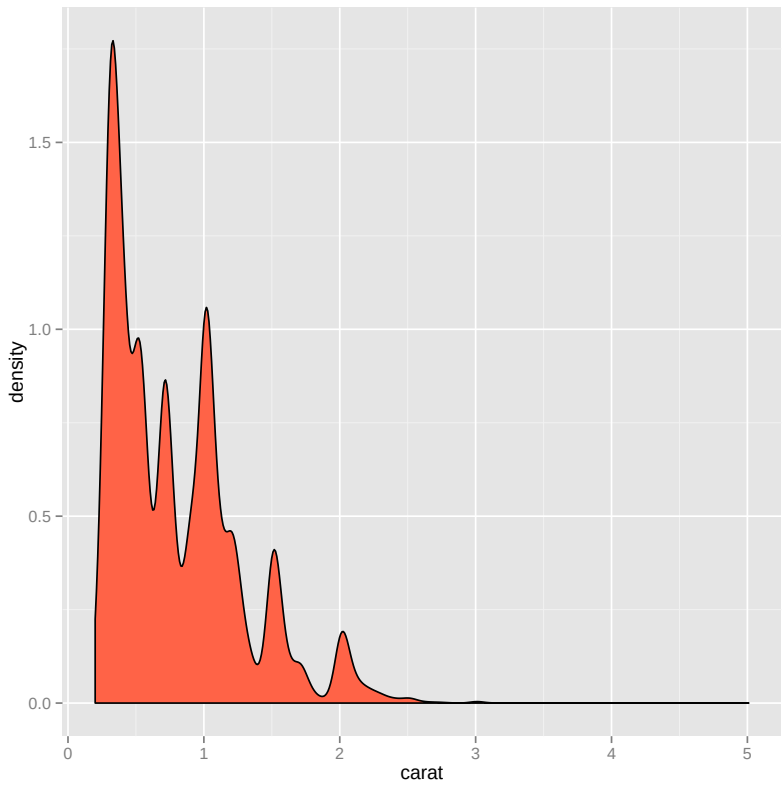

Fig. 3.5. Density plot for diamond prices

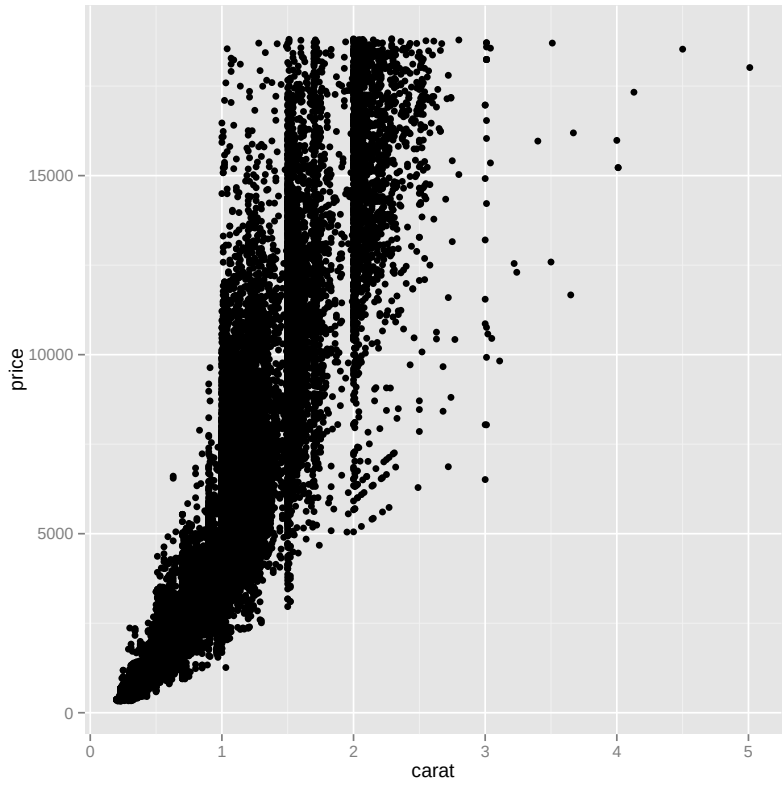

Fig. 3.6. Black and white scatterplot for prices vs. carats

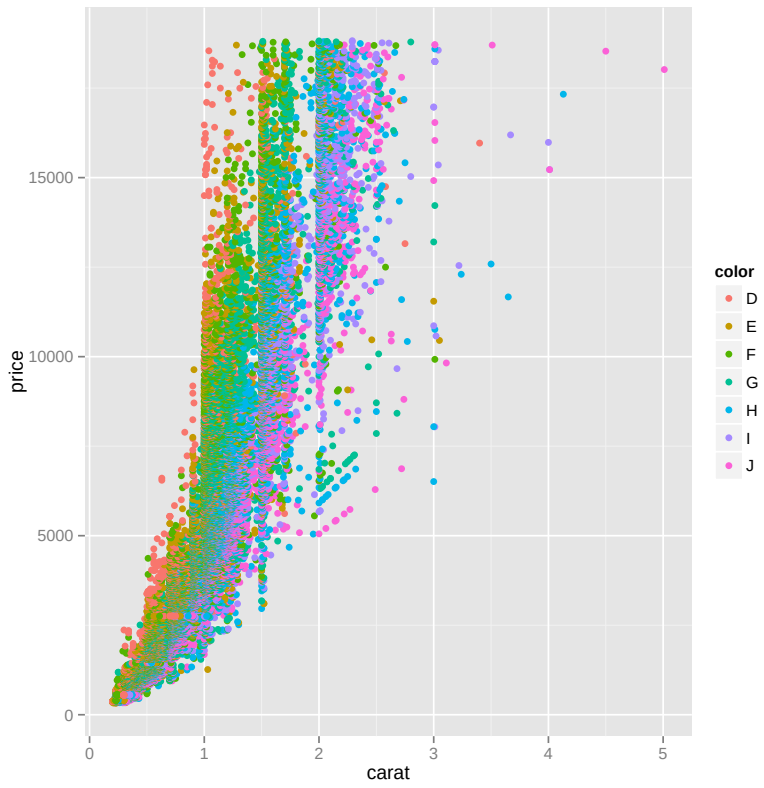

Fig. 3.7. Colored scatterplot for prices vs. carats

```
> pairs(diamonds[c("carat","depth","price")])
```

we obtain the plot array shown in Figure 3.9.

An alternative function of the package `psych` provides even more information. As a response to

```
> pairs.panels(diamonds[c("carat","depth","price")])
```

we obtain the result shown in Figure **??**.

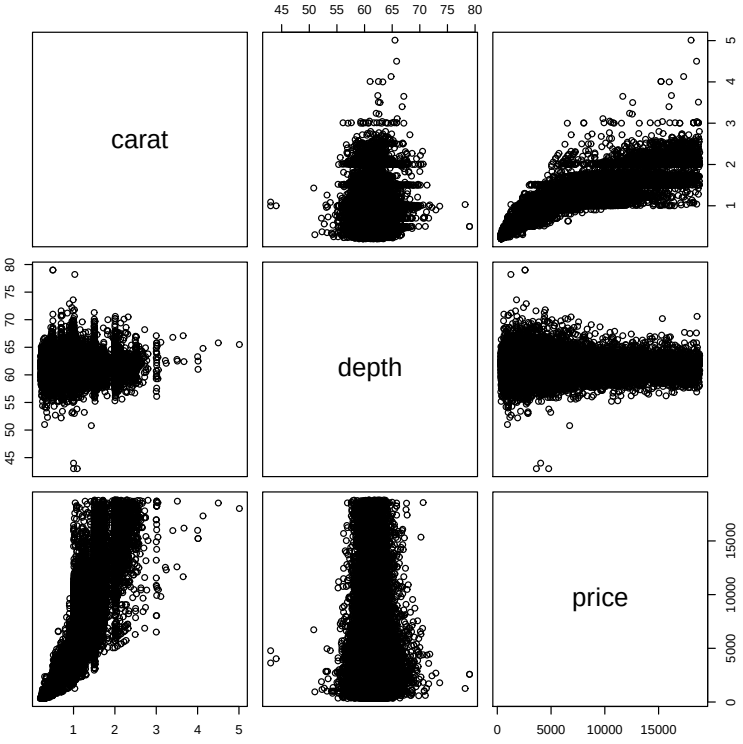

Fig. 3.8. Array of scatterplots obtained with the function `pairs` of the package `psych`.

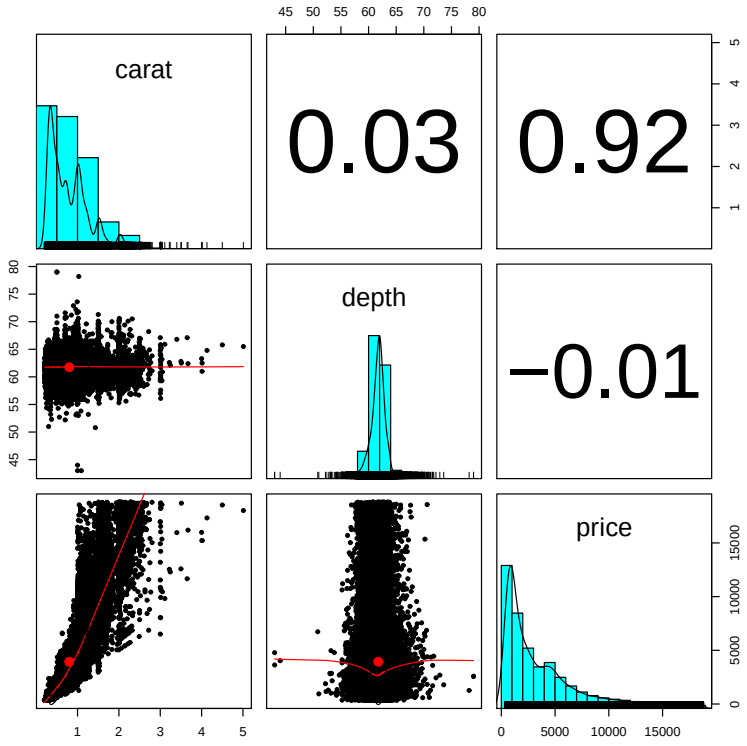

Fig. 3.9. Array of correlation coefficients, histograms, and scatterplots obtained with the function `pairs.panels` of the package `psych`.

3.2 Biplots

Biplots introduced by K. R. Gabriel offer a succinct and powerful way of representing graphically the elements of a matrix using two sets of vectors (hence, the term *biplot*).

Essentially, biplots may be used for representing on the same page two distinct sets of bi-dimensional vectors.

Example 17. Let

$$U = \left\{ \begin{pmatrix} 1.3 \\ 2.1 \end{pmatrix}, \begin{pmatrix} 0.2 \\ -1.5 \end{pmatrix}, \begin{pmatrix} 1.3 \\ 1.3 \end{pmatrix} \right\},$$

and

$$V = \left\{ \begin{pmatrix} 0.1 \\ 2 \end{pmatrix}, \begin{pmatrix} 4.7 \\ 1.1 \end{pmatrix}, \begin{pmatrix} 3.3 \\ -2.1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} \right\}.$$

We will refer to U and V as the set of left vectors and the set of right vectors, respectively.

Let us represent the vectors of U and V in a two-dimensional map shown in Figure 3.10. This can be done in R using the function `biplot` that allows us to place on the same picture both the vectors of U (as points) and the vectors of V as arrows. To prepare the data for the function `biplot` we have to structure it as two-column matrices by writing the vectors of U and V as rows:

```
L <- matrix(c(1.3,0.2,1.3,2.1,-1.5,1.3),ncol=2)
R <- matrix(c(0.1,4.7,3.3,1,2,1.1,-2.1,0.5),ncol=2)
```

Then, we place a call to the function `biplot`:

```
> biplot(L,R,var.axes=TRUE,c(2,4))
```

The first argument and the second argument of `biplot` must be two-column matrices. The parameter `var.axes = TRUE` indicates that the second set of vectors have arrows; finally, `col` is a vector of length 2 that gives the colors for the first and the second set of points including the axes.

Example 18. Biplots can be used to facilitate graphical representation of matrices of rank 2. Let $A \in \mathbb{R}^{3\times 4}$ be the matrix

$$A = \begin{pmatrix} 4 & 6 & -1 & 1 \\ -2 & -1 & 2 & 0 \\ 2 & 9 & 4 & 2 \end{pmatrix}$$

which can be written as a product of two matrices, $A = LR'$, where L and R are the matrices defined in Example 17. Let us write the matrices L and R as

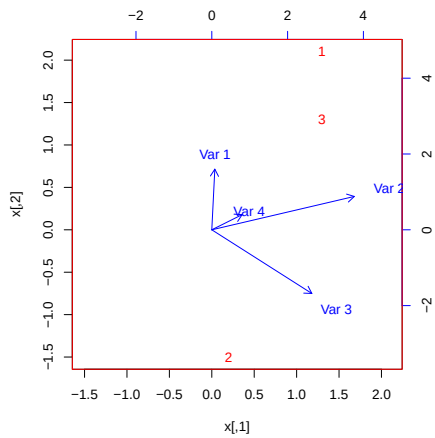

Fig. 3.10. Biplot that represents the sets of vectors that correspond to the rows of L and R

$$L = \begin{pmatrix} \boldsymbol{l}_1' \\ \boldsymbol{l}_2' \\ \boldsymbol{l}_3' \end{pmatrix} \in \mathbb{R}^{3\times 2} \text{ and } R = \begin{pmatrix} \boldsymbol{r}_1' \\ \boldsymbol{r}_2' \\ \boldsymbol{r}_3' \\ \boldsymbol{r}_4' \end{pmatrix} \in \mathbb{R}^{4\times 2},$$

where $\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_4$ all belong to $\mathbb{R}^2$. Thus, the matrix A can be written as

$$A = \begin{pmatrix} \boldsymbol{l}_1'\boldsymbol{r}_1 & \boldsymbol{l}_1'\boldsymbol{r}_2 & \boldsymbol{l}_1'\boldsymbol{r}_3 & \boldsymbol{l}_1'\boldsymbol{r}_4 \\ \boldsymbol{l}_2'\boldsymbol{r}_1 & \boldsymbol{l}_2'\boldsymbol{r}_2 & \boldsymbol{l}_2'\boldsymbol{r}_3 & \boldsymbol{l}_2'\boldsymbol{r}_4 \\ \boldsymbol{l}_3'\boldsymbol{r}_1 & \boldsymbol{l}_3'\boldsymbol{r}_2 & \boldsymbol{l}_3'\boldsymbol{r}_3 & \boldsymbol{l}_3'\boldsymbol{r}_4 \end{pmatrix},$$

that is, $a_{ij} = \boldsymbol{l}_i'\boldsymbol{r}_j = \| \boldsymbol{l}_i' \| \| \boldsymbol{r}_j \| \cos\angle(\boldsymbol{l}_i, \boldsymbol{r}_j)$ for $1 \leqslant i \leqslant 3$ and $1 \leqslant j \leqslant 4$.

When the vectors $\boldsymbol{r}_j$ are unit vectors the elements of A are the projections of the vectors $\boldsymbol{l}$ on the vectors $\boldsymbol{r}$.

This approach can be directly applied to any kind of decomposition $A = PQ'$ of the matrix A, when $P \in \mathbb{R}^{m\times n}$ and $P \in \mathbb{R}^{m\times 2}$ and $Q \in \mathbb{R}^{n\times 2}$.

In general, the rank of a data matrix A is larger than 2. In this case, approximative representations of A can be obtained by using the thin singular value decomposition of matrices.

Let A be a matrix of rank r and let

$$A = UDV' = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i',$$

be the thin SVD, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are matrices of rank r (and, therefore, full-rank matrices) having orthonormal sets of columns. Here $U = (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r)$ and $V = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_r)$.

The matrix D containing singular values can be split between U and V by defining $L = U\sqrt{D}$ and $R = \sqrt{D}V'$.

By Eckhart-Young Theorem the best approximation of A in the sense of the matrix norm $\|\!| \cdot |\!\|_2$ in the class of matrix of rank k is the matrix defined by

$$B(k) = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'.$$

The same matrix $B(k)$ is the best approximation of A in the sense of Frobenius norm. The extent of the deficiency of this approximation is measured by $\| A - B(k) \|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_r^2$. Since $\| A \|_F^2 = \sigma_1^2 + \cdots + \sigma_r^2$, an absolute measure of the quality of the approximation of A by $B(k)$ is

$$q_k = 1 - \frac{\| A - B(k) \|_F^2}{\| A \|_F^2} = \frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

In the special case, $k = 2$, the quality of the approximation is

$$q_2 = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

and it is desirable that this number is as close as one as possible. The rank-2 approximation of A is useful because we can apply biplots to the visualization of A.

Example 19. Let $A \in \mathbb{R}^{5\times 4}$ be the matrix

```
> A
     [,1] [,2] [,3] [,4]
[1,]   -1   -2    0    1
```

```
[2,]   -3   -3   -3    0
[3,]   -7   33  -12    9
[4,]   -2    7   -6   -2
[5,]    1   11    6   11
```

Its singular value decomposition is

```
> S <- svd(A)
> S
$d
[1] 3.932769e+01 1.417821e+01 4.506787e+00 1.927126e-15

$u
            [,1]        [,2]        [,3]        [,4]
[1,]  0.03461325  0.01363572  0.44981361 -0.86565711
[2,]  0.03590229 -0.22915777  0.84346713  0.35308151
[3,] -0.93619495 -0.19064403  0.07026304  0.06591937
[4,] -0.19618241 -0.39940617 -0.26296837 -0.34664351
[5,] -0.28734025  0.86685082  0.11018915 -0.03826402

$v
           [,1]       [,2]       [,3]       [,4]
[1,]  0.1656865 0.25913072 -0.6292577  0.7137464
[2,] -0.9053516 0.07818087 -0.3860947 -0.1586103
[3,]  0.2690136 0.74570386 -0.2517565 -0.5551361
[4,] -0.2837573 0.60882245  0.6257670  0.3965258
```

An examination of its singular values reveals that the rank of A is 3 because the last entry of `d`, $1.927126e-15$ is very small. Let $D \in \mathbb{R}^{3\times 3}$ be the diagonal matrix that contains the non-zero singular values:

```
> D <- matrix(c(3.932769e+01,0,0,0,1.417821e+01,0,0,0,4.506787e+00),ncol=3)
> D
         [,1]     [,2]     [,3]
[1,] 39.32769  0.00000 0.000000
[2,]  0.00000 14.17821 0.000000
[3,]  0.00000  0.00000 4.506787
```

We factor A by allocating the matrix of singular values to the left factor,

```
> L <- S$u[,1:2] %*% D

> L
           [,1]      [,2]
[1,]   1.361259  0.193330
```

```
[2,]   1.411954 -3.249047
[3,] -36.818385 -2.702991
[4,]  -7.715401 -5.662865
[5,] -11.300428 12.290393
```

and by defining the right factor as

```
> R <- S$v[,1:2]
> R
           [,1]       [,2]
[1,]  0.1656865 0.25913072
[2,] -0.9053516 0.07818087
[3,]  0.2690136 0.74570386
[4,] -0.2837573 0.60882245
```

Thus, the rank-2 approximation of A is the matrix B given by

```
> B <- L %*% t(R)
> B
           [,1]      [,2]        [,3]       [,4]
[1,]  0.2756401 -1.217303   0.5103642 -0.2685636
[2,] -0.6079861 -1.532328  -2.0429920 -2.3787450
[3,] -6.8007388 33.122260 -11.9202787  8.8018436
[4,] -2.7457603  6.542422  -6.2983681 -1.2583778
[5,]  1.3124895 11.191734   6.1250240 10.6892462
```

The biplot of A is obtained with

```
> biplot(L,R,var.axes=TRUE,c(2,4))
```

and it is shown in Figure 3.11.

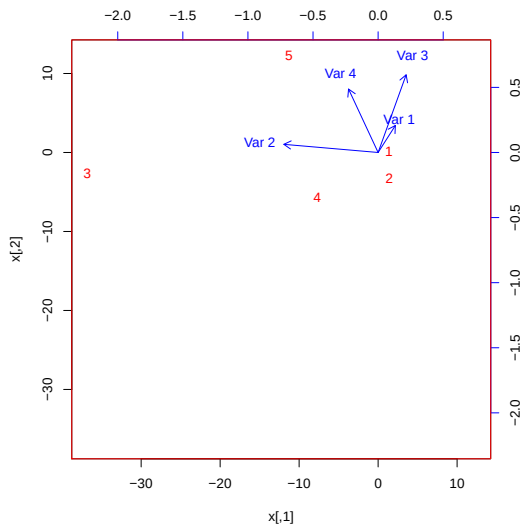

Fig. 3.11. Biplot that represents the sets of vectors that correspond to the rank-2 approximation of matrix A

4

Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction technique that aims to create a few new, uncorrelated linear combinations of the variables of an experiments that "explain" the major parts of the data variability.

The conclusions of a PCA analysis of data are mainly qualitative.

4.1 Principal Components

Let

$$D = \begin{pmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_m' \end{pmatrix} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{m \times n}$$

be a data sample matrix and let $\tilde{D}$ be the corresponding centered matrix. Since $cov(D)$ is a symmetric matrix, there exists a orthogonal matrix R diagonalizes $cov(D)$. Let $R \in \mathbb{R}^{n \times n}$ be the orthogonal matrix that diagonalizes $cov(D)$, that is,

$$R' cov(D) R = \mathsf{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2),$$

or equivalently, $cov(D)R = R\mathsf{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$.

There exists an immediate link between the diagonalization of $cov(D)$ and the SVD decomposition of the centered data matrix $\tilde{D}$. Recall that if $(\sigma, \boldsymbol{r}, \boldsymbol{s})$ is a singular triplet of the matrix $\tilde{D}$, then

$$\tilde{D}\boldsymbol{r} = \sigma\boldsymbol{s}, \tilde{D}'\boldsymbol{r} = \sigma\boldsymbol{r}$$

and therefore, $\tilde{D}'\tilde{D}\boldsymbol{r} = \sigma^2\boldsymbol{r}$ and $\tilde{D}\tilde{D}'\boldsymbol{s} = \sigma^2\boldsymbol{s}$. This shows that $\boldsymbol{r}$, a left singular vector of $\tilde{D}$ is an eigenvector of the covariance matrix $cov(D)$. Thus, the columns of the matrix R that diagonalizes $cov(D)$ are the left singular vectors of $\tilde{D}$.

Let $\boldsymbol{u}$ be a n-dimensional vector of observations that corresponds to variables $\mathcal{V}_1, \ldots, \mathcal{V}_n$. We seek new variables such that the image of an observation vector $\boldsymbol{u} \in \mathbb{R}^n$ of matrix $\tilde{D}'$ becomes $\boldsymbol{z} = R'\boldsymbol{u}$. The new variables are

called *principal components* and the individual transformed observations will be called *scores*.

Let $\tilde{D} = U\mathsf{diag}(\sigma_1, \ldots, \sigma_r)V'$ be the thin SVD of the centered data matrix $\tilde{D} \in \mathbb{R}^{m\times n}$, where $U \in \mathbb{R}^{m\times r}$, and $V \in \mathbb{R}^{r\times n}$ are matrices having orthogonal columns, and $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$ are the non-zero singular values of $\tilde{D}$. For the covariance matrix $cov(D)$ we have

$$cov(D) = \frac{1}{m-1}\tilde{D}'\tilde{D} = \frac{1}{m-1}VZ'U'UZV' = \frac{1}{m-1}VZ'ZV' = \frac{1}{m-1}VZ^2V',$$

where $Z = \mathsf{diag}(\sigma_1, \ldots, \sigma_r)$, due to the orthogonality of the columns of U.

The previous consideration show that

- the columns of V are the eigenvectors of $cov(D)$ that correspond to the top r eigenvalues of $cov(D)$, so they are the first r columns of the matrix R;
- the data matrix $\tilde{D}$ can be written as $\tilde{D} = SV$, where $S = UZ$.

The matrix V is known as the *matrix of loadings*, while the matrix $S = UZ \in \mathbb{R}^{m\times r}$ is known as the *matrix of scores*.

Since the columns of V are orthogonal we also have $S = \tilde{D}V$.

The SVD of $\tilde{D}$ can be written as

$$\tilde{D} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'.$$

This implies $\tilde{D}'\tilde{D}\boldsymbol{v}_i = \sigma_i^2\boldsymbol{v}_i$. Since $\boldsymbol{u}_i'\tilde{D} = \sigma_i\boldsymbol{v}_i'$, it follows that $\boldsymbol{v}_i'$ is a weighted sum of the rows of the matrix $\tilde{D}$. Similarly, $\boldsymbol{u}_i$ are weighted sums of the columns of $\tilde{D}$.

If

$$\tilde{D} = (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r)(\sigma_1\boldsymbol{v}_1' \ \ldots \ \sigma_r\boldsymbol{v}_r)',$$

then

$$\begin{aligned} I_r &= (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r)'(\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r) \\ (n-1)\tilde{D}'\tilde{D} &= (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r)(\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_r)' \end{aligned}$$

The sum of the elements of $\mathsf{diag}(\sigma_1^2, \ldots, \sigma_n^2)$'s main diagonal equals the total variance $\mathsf{tvar}(D)$. The principal directions "explain" the sources of the total variance: sample vectors grouped around $\boldsymbol{r}_1$ explain the largest portion of the variance; sample vectors grouped around $\boldsymbol{r}_2$ explain the second largest portion of the variance, etc.

Example 20. Let us compute the principal components of the `fatprot` data set. Note that the country codes form the first column of `fatprot`. We construct a new data frame referred as `f1`, where the country codes serve as labels for the rows by writing

```
> f1 <- fatprot[,c(2,3)]
> rownames(f1) <- fatprot[,1]
```

Next, we apply the `princomp` function:

```
> p1 <- princomp(f1)
```

which results in the object `p1`, that has the structure

```
> str(p1)
List of 7
 $ sdev    : Named num [1:2] 30.5 11
  ..- attr(*, "names")= chr [1:2] "Comp.1" "Comp.2"
 $ loadings: loadings [1:2, 1:2] 0.374 0.927 -0.927 0.374
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "prot" "fat"
  .. ..$ : chr [1:2] "Comp.1" "Comp.2"
 $ center  : Named num [1:2] 98.2 121.9
  ..- attr(*, "names")= chr [1:2] "prot" "fat"
 $ scale   : Named num [1:2] 1 1
  ..- attr(*, "names")= chr [1:2] "prot" "fat"
 $ n.obs   : int 37
 $ scores  : num [1:37, 1:2] -32.8 34 -26.9 38.6 -55.5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:37] "AL" "AT" "BY" "BE" ...
  .. ..$ : chr [1:2] "Comp.1" "Comp.2"
 $ call    : language princomp(x = f1)
 - attr(*, "class")= chr "princomp"
```

The loadings are:

```
 loadings(p1)
Loadings:
     Comp.1 Comp.2
prot  0.374 -0.927
fat   0.927  0.374

               Comp.1 Comp.2
SS loadings       1.0    1.0
Proportion Var    0.5    0.5
Cumulative Var    0.5    1.0
```

and the scores are

```
 p1$scores
       Comp.1       Comp.2
AL -32.796041 -11.89639300
AT  34.004205   4.28159227
```

```
BY -26.891422   0.19225773
BE  38.607227  16.92327674
BA -55.459440  -9.18155469
BG -26.550694  10.03521992
HR -32.131362  13.17467004
CY  10.609020   3.46592223
CZ  -2.015888   2.68380579
DK  15.832183  -4.13135140
EE -27.818737  -0.18202369
FI   7.290817  -4.34365731
FR  46.092855  -1.62302655
GE -67.173809  -4.20425104
DE  18.954856   6.83445506
GR  29.401182  -8.36009220
HU  18.368269  16.30313582
IS  30.736333 -19.68340329
IE  18.452153 -10.62255755
IT  39.031839  -0.15945444
LV  -9.646715   8.23091998
LT -10.490146 -19.06905482
LU  48.712825  -8.11423270
MK -28.243349  16.90070750
MT  -4.356445 -20.90690835
MD -67.743620  -0.12070895
NL  13.960776   0.50522442
NO  22.680894   2.94644209
PL  -7.563002  -4.94702144
PT  19.932502  -8.94667954
RO  -9.384079 -16.46586164
RU -34.667448  -7.25981718
YU -14.138092  19.35870196
SK -22.679458  19.14639605
SI   9.877234  -0.06458611
ES  31.970822   1.30411767
CH  25.233756  17.99579063
```

Finally, we apply the function `biplot(p1)` which results in Figure 4.1:

Example 21. The data set `USArrests`, a part of the base package of R contains arrest records for all 50 states as well as the percentage of the urban population. A call to the function `class` identifies `USArrests` as a data frame:

```
 class(USArrests)
[1] "data.frame"
```

Columns of this data frame are retrieved by writing

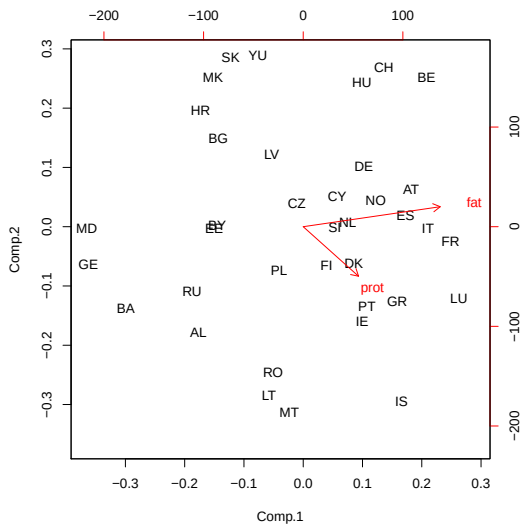

Fig. 4.1. Principal components of fat-protein consumption in European countries

```
 names(USArrests)
[1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

The names of the rows can be obtained with the function `rownames`:

```
 rownames(USArrests)
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
 [5] "California"     "Colorado"       "Connecticut"    "Delaware"
 [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
[17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
[21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
```

```
[29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
[33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
[41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
[45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
```

To apply PCA a scaling is needed, so we write

```
> s <- scale(USArrests)
```

Principal component analysis is performed by the function `pricomp` that returns an object p of the class `princomp`:

```
> p <- princomp(s)
> class(p)
[1] "princomp"
```

The structure of the object `p` is revealed as

```
 str(p)
List of 7
 $ sdev    : Named num [1:4] 1.559 0.985 0.591 0.412
  ..- attr(*, "names")= chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
 $ loadings: loadings [1:4, 1:4] -0.536 -0.583 -0.278 -0.543 0.418 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
  .. ..$ : chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
 $ center  : Named num [1:4] -7.39e-17 9.37e-17 -4.53e-16 1.02e-16
  ..- attr(*, "names")= chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
 $ scale   : Named num [1:4] 1 1 1 1
  ..- attr(*, "names")= chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
 $ n.obs   : int 50
 $ scores  : num [1:50, 1:4] -0.976 -1.931 -1.745 0.14 -2.499 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
  .. ..$ : chr [1:4] "Comp.1" "Comp.2" "Comp.3" "Comp.4"
 $ call    : language princomp(x = s)
 - attr(*, "class")= chr "princomp"
```

The loadings are returned by the function `loadings`:

```
> loadings(p)

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4
Murder   -0.536  0.418 -0.341  0.649
Assault  -0.583  0.188 -0.268 -0.743
```

```
UrbanPop -0.278 -0.873 -0.378  0.134
Rape     -0.543 -0.167  0.818

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
```

The result of the PCA is represented by the biplot shown in Figure 4.2, which is obtained by

```
biplot(p)
```

Example 22. The data set that we are about to analyze originates in a study of the health condition of Boston neighborhoods produced by the Health Department of the City of Boston. The data includes incidence of various diseases and health events that occur in the 16 neighborhoods of the city identified as

Neighborhood	Code	Neighborhood	Code
Allston/Brighton	AB	North Dorchester	ND
Back Bay	BB	North End	NE
Charlestown	CH	Roslindale	RO
East Boston	EB	Roxbury	RX
Fenway	FW	South Boston	SB
Hyde Park	HP	South End	SE
Jamaica Plain	JP	South Dorchester	SD
Mattapan	MT	West Roxbury	WR

The codes of the neighborhoods are grouped into a vector

```
neighborhoods <- c("AB","BB","CH","EB","FW","HP","JP",
"MT","ND","NE","RS","RX","SB","SD","SE","WR")
```

which is used for assigning row names to the data frame `dis` that is used in the analysis.

The diseases and the health conditions are listed as the vector `categories` that will labels the columns of the data frame:

Category	Code	Category	Code
Hepatitis B	HepB	Tuberculosis	TBCD
Hepatitis C	HepC	Live Births	B154
HIV/AIDS	HIVA	Low weight at birth	LBWE
Chlamydia	CHLA	Infant Mortality	INFM
Syphilis	SYPH	Children with Elevated Lead	CELL
Gonorrhea	GONO	Subst. Abuse Treat. Admissions	SATA

This allows us to put together the data frame `dis` shown next:

```
> dis
   HepB HepC HIVA CHLA SYPH GONO TBCD B154 LBWE INFM CELL SATA
AB   38   57   13  168    5   22   20  607   40    7   13  624
BB   17   24   16  179   11   52    8  306   25    0    0  497
CH   10   13    0   46    0    8    0  284   25    0    0  489
EB   12   46   11  150   10   16   17  718   43   10   43 1009
FW   18   19    8  163    9   44    7  125   10    0    0  272
```

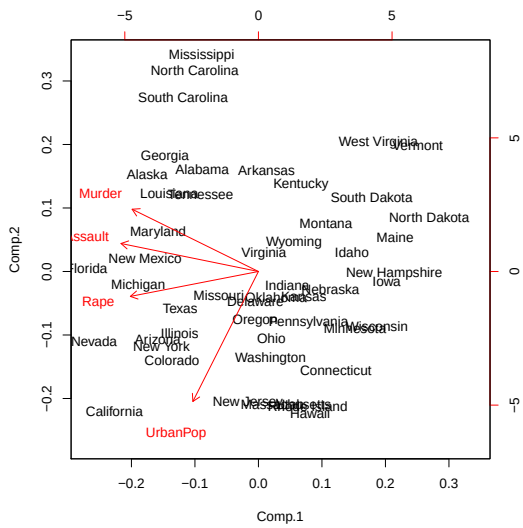

Fig. 4.2. Principal components of the USArrest data set

HP	11	18	8	179	9	25	9	487	46	12	19	2781
JP	6	32	18	213	10	46	6	420	35	5	10	1071
MT	15	24	11	264	14	56	8	285	26	7	25	390
ND	42	76	22	611	15	135	29	1350	168	28	88	1492
NE	0	0	0	0	0	0	0	89	5	0	0	130
RS	13	22	0	115	6	31	0	488	39	6	28	330
RX	21	50	17	477	8	72	8	829	87	27	30	2075
SB	9	52	0	85	7	25	5	403	25	0	18	1335
SD	68	78	23	760	24	176	24	656	67	9	63	1464
SE	51	35	35	124	31	61	11	439	34	0	0	6064
WR	9	10	0	17	0	0	0	419	34	0	11	179

The data is scaled with

```
> s <- scale(dis)
```

and the `princomp` object is obtained:

```
> p <- princomp(s)
```

The stucture of `p` is made visible using

```
> str(p)
```

```
List of 7
 $ sdev    : Named num [1:12] 2.684 1.441 0.912 0.649 0.473 ...
  ..- attr(*, "names")= chr [1:12] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
 $ loadings: loadings [1:12, 1:12] -0.291 -0.321 -0.267 -0.327 -0.243 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:12] "HepB" "HepC" "HIVA" "CHLA" ...
  .. ..$ : chr [1:12] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
 $ center  : Named num [1:12] -1.64e-17 6.94e-18 6.07e-18 8.67e-19 3.24e-17 ...
  ..- attr(*, "names")= chr [1:12] "HepB" "HepC" "HIVA" "CHLA" ...
 $ scale   : Named num [1:12] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "names")= chr [1:12] "HepB" "HepC" "HIVA" "CHLA" ...
 $ n.obs   : int 16
 $ scores  : num [1:16, 1:12] -0.509 1.042 2.857 -0.49 1.801 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "AB" "BB" "CH" "EB" ...
  .. ..$ : chr [1:12] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
 $ call    : language princomp(x = s)
 - attr(*, "class")= chr "princomp"
```

To obtain the loadings we write:

```
> loadings(p)
```

This results in

```
Loadings:
     Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
HepB -0.291 -0.273 -0.264  0.199  0.104  0.530  0.459 -0.254 -0.382
HepC -0.321        -0.159  0.437         0.272 -0.647  0.341         0.166
HIVA -0.267 -0.385  0.143        -0.576 -0.211         0.342 -0.195 -0.308
CHLA -0.327        -0.267 -0.421 -0.114  0.184 -0.198 -0.145  0.113 -0.414
SYPH -0.243 -0.465        -0.133  0.223 -0.453 -0.126        -0.249  0.474
GONO -0.321        -0.346 -0.420  0.109                0.187  0.322  0.100
TBCD -0.322        -0.144  0.493 -0.330 -0.329  0.112 -0.437  0.432
B154 -0.305  0.259  0.318  0.229                0.138  0.289 -0.279 -0.258
LBWE -0.306  0.266  0.273                       0.441  0.333  0.361  0.343
INFM -0.265  0.321  0.383 -0.298 -0.304  0.117 -0.235 -0.432 -0.277  0.344
CELL -0.303  0.290                0.507 -0.434        -0.157 -0.205 -0.326
```

```
SATA -0.142 -0.470  0.585         0.335  0.209 -0.156 -0.220  0.353 -0.237
     Comp.11 Comp.12
HepB -0.108
HepC -0.187
HIVA -0.349
CHLA  0.321   0.494
SYPH  0.325   0.206
GONO         -0.656
TBCD  0.137
B154  0.601  -0.267
LBWE -0.159   0.408
INFM -0.125  -0.177
CELL -0.446
SATA

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.083  0.083  0.083  0.083  0.083  0.083  0.083  0.083  0.083
Cumulative Var  0.083  0.167  0.250  0.333  0.417  0.500  0.583  0.667  0.750
               Comp.10 Comp.11 Comp.12
SS loadings      1.000   1.000   1.000
Proportion Var   0.083   0.083   0.083
Cumulative Var   0.833   0.917   1.000
```

The results can be visualized using

```
> biplot(p)
```

and yield the graphical representation from Figure 4.3.

The scores of variables relative to the principal components are obtained by examining `p$scores`:

```
> p$scores
       Comp.1      Comp.2        Comp.3      Comp.4        Comp.5      Comp.6
AB -0.5085882  0.23072335 -4.008224e-01  1.67763097 -0.8190283917  0.56642546
BB  1.0422861 -0.77211613 -6.151926e-01 -0.21556817 -0.6485752045 -0.23514907
CH  2.8565744  0.50076440 -3.630303e-02 -0.02026970  0.1596485885  0.49310899
EB -0.4896467  0.82410031  4.339108e-01  1.03454089 -0.0006699524 -0.86286927
FW  1.8010114 -0.57048155 -9.907785e-01 -0.26856720 -0.3520192539 -0.04134139
HP  0.4802148 -0.02526842  1.274354e+00 -0.29100091  0.1858231908 -0.08483288
JP  0.6237991 -0.31469275  1.469908e-01 -0.39932780 -0.5805264140 -0.35083371
MT  0.4745792 -0.13479898 -6.436400e-01 -0.69584218 -0.0913477252 -0.63312916
ND -6.5034700  2.20754750  6.605574e-01  0.09587301  0.1119336952 -0.32514060
NE  3.7082730  0.41804515 -1.779483e-01 -0.31813549 -0.0791396832 -0.08397898
RS  1.4321192  0.99691484  5.947052e-06 -0.32278874  0.7748890804  0.04466975
RX -2.2388386  1.13042682  1.313769e+00 -1.01507260 -0.5304667442  0.77884244
SB  1.3490577  0.26170528 -1.684187e-01  0.75128191  0.8279087192  0.22381039
SD -5.1815779 -0.96937971 -2.320808e+00 -0.35479234  0.4155555892  0.29227541
```

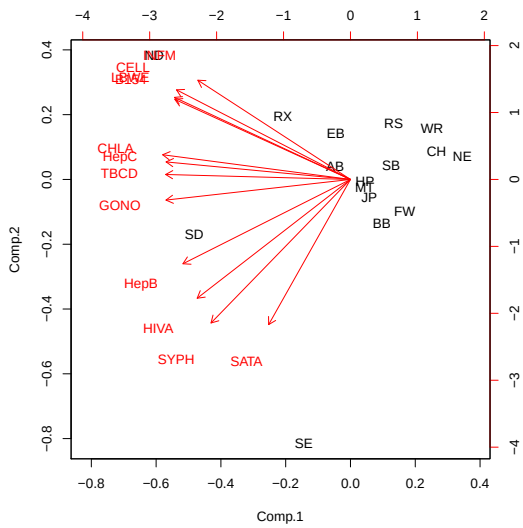

Fig. 4.3. Principal components of incidence of diseases in Boston neighborhoods

```
SE -1.5470288 -4.69574695  1.383851e+00  0.22074136  0.2771234237  0.03849913
WR  2.7012350  0.91225685  1.404722e-01  0.12129701  0.3488910822  0.17964348
        Comp.7      Comp.8       Comp.9      Comp.10      Comp.11       Comp.12
AB  0.09825324 -0.22520221 -0.127491071  0.067976782 -0.020297053  0.0025013114
BB  0.19087038  0.40924448  0.090290109  0.043316652  0.128225066 -0.0425442205
CH  0.33139626  0.07757677  0.120382476 -0.055282844  0.010492757  0.0328460307
EB -0.37692480 -0.20862902 -0.231975979 -0.162285599  0.026202258 -0.0129968454
FW  0.08360278 -0.19022238  0.139520376  0.203825809  0.030648418 -0.0088029984
HP -0.06144973 -0.70212058  0.339422467 -0.074186122  0.105160126 -0.0081502321
JP -0.35680261  0.60325268  0.083003097 -0.184804923 -0.044572371  0.0023532491
MT -0.18110324 -0.25242067 -0.207803525  0.185088441 -0.030148543  0.0826548062
ND  0.59568222  0.23733816  0.236481506  0.126518008 -0.042281228  0.0008062188
NE  0.20430032 -0.28503700  0.143873766 -0.070998961 -0.184143318 -0.0469753605
```

```
RS  0.14343189  0.10416653 -0.430652352  0.119037486  0.025223042 -0.0702244031
RX -0.55315578 -0.03356923 -0.205895022  0.001439052 -0.001427040  0.0016547888
SB -0.87985197  0.35153510  0.281996313  0.129425789 -0.010066987  0.0110503512
SD -0.09619151 -0.19440008 -0.002535052 -0.197943853  0.004576351 -0.0066213763
SE  0.24695354  0.08950775 -0.094236800  0.038456169 -0.041110982  0.0022813426
WR  0.61098900  0.21897969 -0.134380309 -0.169581885  0.043519504  0.0601673376
```

The matrix `score` contains the data obtained by transforming the original data into the space of the principal components.

4.2 A Geometric Perspective of PCA

Next, we present a geometric point of view of principal component analysis.

Let $\boldsymbol{t} \in \mathbb{R}^n$ be a unit vector. The projection of a vector $\boldsymbol{w} \in \mathbb{R}^n$ on the subspace $\langle \boldsymbol{t} \rangle$ generated by $\boldsymbol{t}$ is given by

$$\mathsf{proj}_{\langle \boldsymbol{t} \rangle}(\boldsymbol{w}) = \boldsymbol{t}\boldsymbol{t}'\boldsymbol{w}.$$

To simplify the notation we shall write $\mathsf{proj}_{\boldsymbol{t}}$ instead of $\mathsf{proj}_{\langle \boldsymbol{t} \rangle}$. Let $\tilde{D} \in \mathbb{R}^{m \times n}$ be a centered sample matrix that corresponds to a sequence of experiments $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m)$, that is

$$\tilde{W} = \begin{pmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_m' \end{pmatrix}.$$

We seek to evaluate the inertia $I_{\mathbf{0}}(\mathsf{proj}_{\boldsymbol{t}}(\tilde{D}'))$ on the subspace generated by the unit vector $\boldsymbol{t} \in \mathbb{R}^n$. Since $\tilde{D}' = (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_m)$, by the definition of the inertia, we have:

$$\begin{aligned}
I_{\mathbf{0}}(\mathsf{proj}_{\boldsymbol{t}}(\tilde{D}')) &= \sum_{j=1}^{m} \| \boldsymbol{t}\boldsymbol{t}'\boldsymbol{u}_j \|_2^2 \\
&= \sum_{j=1}^{m} \boldsymbol{u}_j'\boldsymbol{t}\boldsymbol{t}'\boldsymbol{t}\boldsymbol{t}'\boldsymbol{u}_j \\
&= \sum_{j=1}^{m} \boldsymbol{u}_j'\boldsymbol{t}\boldsymbol{t}'\boldsymbol{u}_j \\
&\quad \text{(because } \boldsymbol{t}'\boldsymbol{t} = 1\text{)} \\
&= \sum_{j=1}^{m} \boldsymbol{t}'\boldsymbol{u}_j\boldsymbol{u}_j'\boldsymbol{t} \\
&\quad \text{(because both } \boldsymbol{u}_j'\boldsymbol{t} \text{ and } \boldsymbol{t}'\boldsymbol{u}_j \text{ are scalars)} \\
&= \boldsymbol{t}'X'X\boldsymbol{t}.
\end{aligned}$$

The necessary condition for the existence of extreme values of this inertia as a function of $\boldsymbol{t}$ is

$$\begin{aligned}\mathsf{grad}\ \left(I_{\mathbf{0}}(\mathsf{proj}_{\boldsymbol{t}}(\tilde{D}')) + \lambda(1-\boldsymbol{t}'\boldsymbol{t})\right) &= \mathsf{grad}\ \left(\boldsymbol{t}'\tilde{D}'\tilde{D}\boldsymbol{t} + \lambda(1-\boldsymbol{t}'\boldsymbol{t})\right)\\ &= 2\tilde{D}'\tilde{D}\boldsymbol{u} - 2\lambda\boldsymbol{t} = \mathbf{0},\end{aligned}$$

where λ is a Lagrange multiplier. This implies $\tilde{D}'\tilde{D}\boldsymbol{t} = \lambda\boldsymbol{t}$. In other words, to achieve extreme values of the inertia $I_{\mathbf{0}}(\mathsf{proj}_{\boldsymbol{t}}(\tilde{D}'))$, $\boldsymbol{t}$ must be chosen as a eigenvector of the covariance matrix of $\tilde{D}$.

The principal directions of a data sample matrix $\tilde{D} \in \mathbb{R}^{m\times n}$ can be obtained directly from the data sample matrix D by applying Courant-Fisher Theorem.

Suppose that the eigenvalues of $\tilde{D}'\tilde{D}$ are the numbers $\lambda_1 \geqslant \cdots \geqslant \lambda_n$. The first principal direction $\boldsymbol{t}_1$ of D which corresponds to the largest eigenvalue of $\tilde{D}'\tilde{D}$ is

$$\begin{aligned}\boldsymbol{t}_1 &= \arg\max_{\boldsymbol{t}} \left\{\boldsymbol{t}'\tilde{D}\tilde{D}'\boldsymbol{t} \mid \boldsymbol{t} \in \mathbb{R}^n, \| \boldsymbol{t} \|_2 = 1\right\}\\ &= \arg\max_{\boldsymbol{t}} \left\{\| \tilde{D}'\boldsymbol{t} \|_2^2 \mid \| \boldsymbol{t} \|_2 = 1\right\}.\end{aligned}$$

Suppose that we computed the principal directions $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_k$ of $\tilde{D}$. Then, $\boldsymbol{t}_{k+1} \in \mathbb{R}^n$ is a unit vector $\boldsymbol{t}$ that maximizes

$$\boldsymbol{t}'\tilde{D}\tilde{D}'\boldsymbol{t} = \| \tilde{D}'\boldsymbol{t} \|_2^2$$

and belongs to the subspace orthogonal to the subspace generated by the first k principal directions of $\tilde{D}$, that is,

$$\boldsymbol{t}_{k+1} = \arg\max_{\boldsymbol{t}} \left\{\| \tilde{D}'\boldsymbol{t} \|_2^2 \mid \boldsymbol{t} \in \mathbb{R}^n, \| \boldsymbol{t} \|_2 = 1, \boldsymbol{t} \in \langle \boldsymbol{t}_1, \ldots, \boldsymbol{t}_k\rangle^{\perp}\right\}.$$

Note that for every vector $\boldsymbol{z} \in \mathbb{R}^n$ we have

$$(I - \sum_{j=1}^{k} \boldsymbol{t}_j\boldsymbol{t}_j')\boldsymbol{z} = \boldsymbol{z} - \mathsf{proj}_{\langle \boldsymbol{t}_1,\ldots,\boldsymbol{t}_k\rangle}\boldsymbol{z} \in \langle \boldsymbol{t}_1, \ldots, \boldsymbol{t}_k\rangle^{\perp}.$$

Therefore, $\boldsymbol{x} \in \langle \boldsymbol{t}_1, \ldots, \boldsymbol{t}_k\rangle^{\perp}$, is equivalent to $\boldsymbol{x} = (I - \sum_{j=1}^{k} \boldsymbol{t}_j\boldsymbol{t}_j')\boldsymbol{x}$. Thus, we can write

$$\boldsymbol{t}_{k+1} = \arg\max_{\boldsymbol{t}} \left\{\left\|\tilde{D}(I - \sum_{j=1}^{k} \boldsymbol{t}_j\boldsymbol{t}_j')\boldsymbol{t}\right\|_2 \mid \| \boldsymbol{t} \|_2 = 1\right\},$$

for $0 \leq k \leq n-1$. This technique allows finding the principal directions of $\tilde{D}$ by solving a sequence of optimization problems involving the matrix $\tilde{D}$.

An r-dimensional subspace $\mathcal{T}$ of $\mathbb{R}^n$ is spanned by r linearly independent vectors. Let $T \in \mathbb{R}^{n\times r}$ be an matrix,

$$T = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_r)$$

whose columns $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_r$ form an orthonormal basis of the subspace. In other words we have $T'T = I_r$. Also, note that $TT' \in \mathbb{R}^{n\times n}$.

The projection of a vector $\boldsymbol{x} \in \mathbb{R}^n$ on $\mathcal{T}$ is given by

$$\mathsf{proj}_{\mathcal{T}}(\boldsymbol{x}) = (\boldsymbol{t}_1'\boldsymbol{x})\boldsymbol{t}_1 + \cdots + (\boldsymbol{t}_r'\boldsymbol{x})\boldsymbol{t}_r = TT'\boldsymbol{x}.$$

Therefore, for a matrix $X \in \mathbb{R}^{n\times p}$, the projection of its columns on $\mathcal{T}$ consist of the columns of the matrix $TT'X$. The difference $M = X - TT'X \in \mathbb{R}^{n\times p}$ is the *residual matrix*.

Note that $M = (I_n - TT')X \in \mathbb{R}^{n\times p}$ and that

$$(I_n - TT')(I_n - TT') = I_n - 2TT' + T(T'T)T' = I_n - TT',$$

which means that I_nTT' is both idempotent and symmetric.

We seek a subspace $\mathcal{T}$ such that $\| M \|_F$ is minimized. Since

$$\| M \|_F^2 = \mathsf{trace}(MM') = \mathsf{trace}(M'M).$$

and

$$\begin{aligned}\mathsf{trace}(M'M) &= \mathsf{trace}(X'(I_n - TT')(I_n - TT')X)\\ &= \mathsf{trace}(X'(I_n - TT')X) = \mathsf{trace}(X'X) - \mathsf{trace}(X'TT'X),\end{aligned}$$

the minimization of $\mathsf{trace}(M'M)$ amounts to maximizing $\mathsf{trace}(X'TT'X) = \mathsf{trace}(TT'XX') = \mathsf{trace}(T'XX'T)$. If $XX' \in \mathbb{R}^{n\times n}$ can be diagonalized as $XX' = D\Lambda D'$, then we need to maximize $\mathsf{trace}(T'D\Lambda D'T) = \mathsf{trace}(P'\Lambda P)$, where $P = D'T \in$ is orthonormal matrix.

Note that if P is an orthonormal matrix that achieves this maximum, and Q is an arbitrary orthonormal matrix, then then PQ also achieves the maximum because

$$\mathsf{trace}((PQ)'\Lambda(PQ)) = \mathsf{trace}(\Lambda(PQ)(PQ)') = \mathsf{trace}(\Lambda PQQ'P') = \mathsf{trace}(P'\Lambda P).$$

The expression to be minimized is

$$\begin{aligned}\mathsf{trace}(P'\Lambda P) &= \sum_i \sum_j \sum_k (P')_{ij}\Lambda_{jk}(P)_{ki}\\ &= \sum_i \sum_j (P')_{ij}\lambda_j(P)_{ji}\\ &= \sum_i \sum_j \lambda_j(P)_{ji}^2\\ &= \sum_j \sum_i \lambda_j(P)_{ji}^2\\ &= \sum_j \sum_i \lambda_j f_{ji},\end{aligned}$$

where $f_{ji} = (P_{ji})^2$. The orthonormality of P implies that $0 \leqslant f_{ji} \leqslant 1$, $\sum_{i=1}^{n} f_{ij} = 1$.

Thus, the problem of the minimization of the total values of the residues has seen reduced to a linear programming problem in the indeterminates f_{ij}:

- maximize $\sum_j \sum_i \lambda_j f_{ji}$, subjected to
- – $0 \leqslant f_{ij} \leqslant 1$;
  – $\sum_{i=1}^{n} f_{ij} = 1$ for $1 \leqslant j \leqslant r$;
  – $0 \leqslant \sum_{j=1}^{r} f_{ij} \leqslant 1$.

5

Least Squares Approximation and Data Mining

5.1 Introduction

The least square method is used in data mining as a method of estimating the parameters of a model by adopting the values that minimize the sum of the squared differences between the predicted and the observed values of data. This estimation process is also known as *regression*, and several types of regression exist depending on the nature of the assumed model of dependency between the predicted and the observed data.

5.2 Linear Regression

The aim of *linear regression* is to explore the existence of a linear relationship between the outcome of an experiment and values of variables that are measured during the experiment. As we saw, experimental data often is presented as a data sample matrix $D \in \mathbb{R}^{m \times n}$, where m is the number of experiments and n is the number of variables measured. The results of the experiments are the components of a vector

$$\boldsymbol{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}.$$

Linear regression amounts to determining $\boldsymbol{r} \in \mathbb{R}^n$ such that $D\boldsymbol{r} + k\mathbf{1}_m = \boldsymbol{b}$, where $k \in \mathbb{R}$. Solving this system allows us to express the components of $\boldsymbol{r}$ as linear combinations of the values of the variables. Unfortunately, since m is usually much larger than n, this system is overdetermined and, in general, is inconsistent.

To simplify the presentation we will assume initially that $k = 0$. Note that the more general case (when $\boldsymbol{k} \neq 0$) can be dealt with by adding $\mathbf{1}_n$ as the first column to D and k as the first component to $\boldsymbol{r}$. For obvious reasons k is referred to as the *intercept*.

The variables $\mathcal{V}_1, \ldots, \mathcal{V}_n$ that correspond to the columns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ of the matrix D are referred to in this context as *regressors*; the linear combination $r_1\boldsymbol{v}_1 + \cdots + r_n\boldsymbol{v}_n$ is the *regression of* $\boldsymbol{b}$ *onto the regressors* $\mathcal{V}_1, \ldots, \sqsubseteq_n$.

If the linear system $D\boldsymbol{r} = \boldsymbol{b}$ has no solution, the "next best thing" is to find a vector $\boldsymbol{c} \in \mathbb{R}^n$ such that

$$\| D\boldsymbol{c} - \boldsymbol{b} \|_2 \leq \| D\boldsymbol{w} - \boldsymbol{b} \|_2$$

for every $\boldsymbol{w} \in \mathbb{R}^n$, an approach known as *the least square method*. We will refer to the triple $(D, \boldsymbol{r}, \boldsymbol{b})$ as an *instance of the least square problem*.

Note that $D\boldsymbol{r} \in \mathsf{range}(D)$ for any $\boldsymbol{r} \in \mathbb{R}^n$. Thus, solving this problem amounts to finding a vector $D\boldsymbol{r}$ in the subspace $\mathsf{range}(D)$ such that $D\boldsymbol{r}$ is as close to $\boldsymbol{b}$ as possible.

Let $D \in \mathbb{R}^{m\times n}$ be a full-rank matrix such that $m > n$, so $\mathsf{rank}(D) = n$. The symmetric square matrix $D'D \in \mathbb{R}^{n\times n}$ has the same rank n as the matrix D. Therefore, the system $(D'D)\boldsymbol{r} = D'\boldsymbol{b}$ a unique solution $\boldsymbol{r} = (D'D)^{-1}D'\boldsymbol{b}$. Moreover, $D'D$ is positive definite because $\boldsymbol{r}'D'D\boldsymbol{r} = (D\boldsymbol{r})'D\boldsymbol{r} = \| D\boldsymbol{r} \|_2^2 > 0$ for $\boldsymbol{r} \neq \boldsymbol{0}$.

Theorem 9. *Let $D \in \mathbb{R}^{m\times n}$ be a full-rank matrix such that $m > n$ and let $\boldsymbol{b} \in \mathbb{R}^m$. The unique solution of the system $(D'D)\boldsymbol{r} = D'\boldsymbol{b}$ equals the projection of the vector $\boldsymbol{b}$ on the subspace $\mathsf{range}(D)$.*

Proof. The n columns of the matrix $D = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n)$ constitute a basis of the subspace $\mathsf{range}(D)$. Therefore, we seek the projection $\boldsymbol{c}$ of $\boldsymbol{b}$ on $\mathsf{range}(D)$ as a linear combination $\boldsymbol{c} = D\boldsymbol{t}$, which allows us to reduce this problem to a minimization of the function

$$\begin{aligned} f(\boldsymbol{t}) &= \| D\boldsymbol{t} - \boldsymbol{b} \|_2^2 \\ &= (D\boldsymbol{t} - \boldsymbol{b})'(D\boldsymbol{t} - \boldsymbol{b}) = (\boldsymbol{t}'D' - \boldsymbol{b}')(D\boldsymbol{t} - \boldsymbol{b}) \\ &= \boldsymbol{t}'D'D\boldsymbol{t} - \boldsymbol{b}'D\boldsymbol{t} - \boldsymbol{t}'D'\boldsymbol{b} + \boldsymbol{b}'\boldsymbol{b}. \end{aligned}$$

The necessary condition for the minimum is

$$(\nabla f)(\boldsymbol{t}) = 2D'D\boldsymbol{t} - 2D'\boldsymbol{b} = 0,$$

which implies $D'D\boldsymbol{t} = D'\boldsymbol{b}$.

The linear system $(D'D)\boldsymbol{t} = D'\boldsymbol{b}$ is known as the *system of normal equations* of D and $\boldsymbol{b}$.

Suppose now that $D \in \mathbb{R}^{m\times n}$ has rank k, where $k < \min\{m, n\}$, and $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ are orthonormal matrices such that D can be factored as $B = UMV'$, where

$$M = \begin{pmatrix} R & O_{k,n-k} \\ O_{m-k,k} & O_{m-k,n-k} \end{pmatrix} \in \mathbb{R}^{m\times n},$$

$R \in \mathbb{R}^{k\times k}$, and $\mathsf{rank}(R) = k$.

For $\boldsymbol{b} \in \mathbb{R}^m$ define $\boldsymbol{c} = U'\boldsymbol{b} \in \mathbb{R}^m$ and let

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix},$$

where $\boldsymbol{c}_1 \in \mathbb{R}^k$ and $\boldsymbol{c}_2 \in \mathbb{R}^{m-k}$. Since $\mathsf{rank}(R) = k$, the linear system $R\boldsymbol{z} = \boldsymbol{c}_1$ has a unique solution $\boldsymbol{z}_1$.

Theorem 10. *All vectors $\boldsymbol{r}$ that minimize $\| B\boldsymbol{r} - \boldsymbol{b} \|_2$ have the form*

$$\boldsymbol{r} = V \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{w} \end{pmatrix},$$

for an arbitrary $\boldsymbol{w}$.

Proof. We have

$$\begin{aligned} \| B\boldsymbol{r} - \boldsymbol{b} \|_2^2 &= \| UMV'\boldsymbol{r} - UU'\boldsymbol{b} \|_2^2 \\ &= \| U(MV'\boldsymbol{r} - U'\boldsymbol{b}) \|_2^2 = \| MV'\boldsymbol{r} - U'\boldsymbol{b} \|_2^2 \\ &\quad \text{(because multiplication by an orthonormal matrix} \\ &\quad \text{is norm-preserving)} \\ &= \| MV'\boldsymbol{r} - \boldsymbol{c} \|_2^2 = \| M\boldsymbol{y} - \boldsymbol{c} \|_2^2 \\ &= \| R\boldsymbol{z} - \boldsymbol{c}_1 \|_2^2 + \| \boldsymbol{c}_2 \|_2^2, \end{aligned}$$

where $\boldsymbol{z}$ consists of the first r components of $\boldsymbol{y}$. This shows that the minimal value of $\| B\boldsymbol{r} - \boldsymbol{b} \|_2^2$ is achieved by the solution of the system $R\boldsymbol{z} = \boldsymbol{c}_1$ and is equal to $\| \boldsymbol{c}_2 \|_2^2$. Therefore, the vectors $\boldsymbol{r}$ that minimize $\| B\boldsymbol{r} - \boldsymbol{b} \|_2^2$ have the form $\begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{w} \end{pmatrix}$ for an arbitrary $\boldsymbol{w} \in \mathbb{R}^{n-r}$.

Instead of the Euclidean norm we can use the $\| \cdot \|_\infty$. Note that we have $t = \| B\boldsymbol{r} - \boldsymbol{b} \|_\infty$ if and only if $-t\mathbf{1} \leq B\boldsymbol{r} - \boldsymbol{b} \leq t\mathbf{1}$, so finding $\boldsymbol{r}$ that minimizes $\| \cdot \|_\infty$ amounts to solving a linear programming problem: minimize t subjected to the restrictions $-t\mathbf{1} \leq B\boldsymbol{r} - \boldsymbol{b} \leq t\mathbf{1}$.

Similarly, we can use the norm $\| \cdot \|_p$. If $\boldsymbol{y} = B\boldsymbol{r} - \boldsymbol{b}$, then we need to minimize $\| \boldsymbol{y} \|_p^p = |y_1|^p + \cdots + |y_m|^p$, subjected to the restrictions $-\boldsymbol{y} \leq A\boldsymbol{r} - \boldsymbol{b} \leq \boldsymbol{y}$.

Example 23. In Figure 5.1 we represent (using the function `plot` of MATLAB), the number of calories consumed by a person per day vs. the gross national product per person in European countries starting from the table included below.

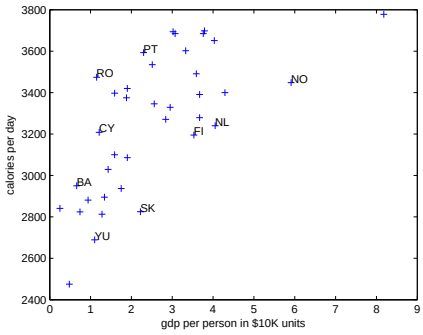

Fig. 5.1. Calories vs. GDP in 10K units per person in Europe.

ccode	gdp	cal	ccode	gdp	cal
'AL'	0.74	2824.00	'IT'	3.07	3685.00
'AT'	4.03	3651.00	'LV'	1.43	3029.00
'BY'	1.34	2895.00	'LT'	1.59	3397.00
'BE'	3.79	3698.00	'LU'	8.18	3778.00
'BA'	0.66	2950.00	'MK'	0.94	2881.00
'BG'	1.28	2813.00	'MT'	2.51	3535.00
'HR'	1.75	2937.00	'MD'	0.25	2841.00
'CY'	1.21	3208.00	'NL'	4.05	3240.00
'CZ'	2.56	3346.00	'NO'	5.91	3448.00
'DK'	3.67	3391.00	'PL'	1.88	3375.00
'EE'	1.90	3086.00	'PT'	2.30	3593.00
'FI'	3.53	3195.00	'RO'	1.15	3474.00
'FR'	3.33	3602.00	'RU'	1.59	3100.00
'GE'	0.48	2475.00	'YU'	1.10	2689.00
'DE'	3.59	3491.00	'SK'	2.22	2825.00
'GR'	3.02	3694.00	'SI'	2.84	3271.00
'HU'	1.90	3420.00	'ES'	2.95	3329.00
'IS'	3.67	3279.00	'CH'	4.29	3400.00
'IE'	3.76	3685.00			

We seek to approximate the calorie intake as a linear function of the gdp of the form

$$\text{cal} = r_1 + r_2 \text{ gdp}.$$

This amounts to solving a linear system that consists of 37 equations and two unknowns:

$$\begin{aligned} r_1 + 0.74 r_2 &= 2824 \\ &\vdots \\ r_1 + 4.29 r_2 &= 3400 \end{aligned}$$

and, clearly such a system is inconsistent.

To accommodate a constant term r_1 we construct the matrix D starting from the data frame `calgdp` by a combining a column that consists of 1s with the `calgdp[,2]` using the column binding function of **R** and the function `ones` of the `matlab` package, `cbind`:

```
> D<- cbind(ones(37,1),calgdp[,2])
```

Thus, we work with matrix $D \in \mathbb{R}^{37\times 2}$ given by

$$D = \begin{pmatrix} 1 & 0.74 \\ \vdots & \vdots \\ 1 & 4.29 \end{pmatrix}$$

whose second column consists of the countries' gross domestic products in \$10K units.

The matrix $C = D'D$ is

$$C = \begin{pmatrix} 37.0000 & 94.4600 \\ 94.4600 & 333.6592 \end{pmatrix}.$$

and the vector $\boldsymbol{c}$ is given by $\boldsymbol{c} = D'\boldsymbol{b}$.

Solving the normal system $C\boldsymbol{t} = \boldsymbol{c}$ in **R** can be done using the function

```
solve(C,c)
```

which yields

```
> C <- t(D) %*% D
> c <- t(D) %*% b
> solve(C,c)
          [,1]
[1,] 2894.1643
[2,]  142.3451
```

Thus, the regression line is

$$\mathsf{cal} = 142.3 * \mathsf{gdp} + 2894.2,$$

shown in Figure 5.2. This line can be drawn by writing

```
qplot(x = gdp,y=cal,data=calgdp)+geom_abline(intercept=2894,slope=142,color="red")
```

and is represented in Figure 5.2

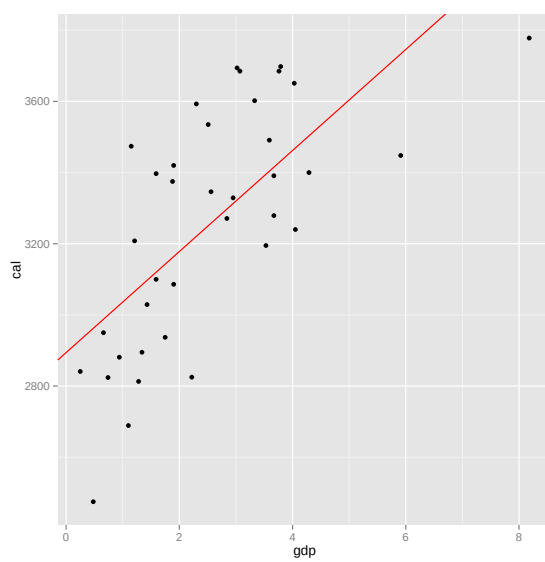

Fig. 5.2. Regression Line

Another approach is offered by the `lm` (linear model) function of **R** :

```
> cg <- calgdp[,2:3]
> calLM <- lm(cal ~ gdp,data=cg)
```

The function `lm` returns an object whose structure is given next:

```
> calLM

Call:
lm(formula = cal ~ gdp, data = cg)

Coefficients:
(Intercept)          gdp
     2894.2        142.3

> summary(calLM)

Call:
lm(formula = cal ~ gdp, data = cg)

Residuals:
    Min      1Q  Median      3Q     Max
-487.49 -189.91  -27.42  233.83  416.14

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2894.16      76.91  37.629  < 2e-16 ***
gdp           142.35      25.61   5.558 2.96e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 246.3 on 35 degrees of freedom
Multiple R-squared:  0.4688,    Adjusted R-squared:  0.4536
F-statistic: 30.89 on 1 and 35 DF,  p-value: 2.957e-06
```

In Example 23 we had only one regressor, namely `gdp`. In many practical problems we need to deal with several regressors.

Example 24. Consider again the `diamonds` data set of the `ggplot2` package. To find a regression of `price` on the regressors `carat`, `cut`, `clarity`, and `color`. amounts to seeking an approximate solution of the system

$$(\mathbf{1}\ diamonds[c(1:4)])\boldsymbol{r} = diamonds[7].$$

In **R** we use the function `lm` as follows:

```
priceRegr <- lm(price ~ carat + cut + clarity + color,data = diamonds)
```

The structure of the object `priceRegr` is obtained by

```
> summary(priceRegr)

Call:
lm(formula = price ~ carat + cut + clarity + color, data = diamonds)

Residuals:
     Min       1Q   Median       3Q      Max 
-16813.5   -680.4   -197.6    466.4  10394.9 

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)    
(Intercept) -3710.603     13.980 -265.414  < 2e-16 ***
carat        8886.129     12.034  738.437  < 2e-16 ***
cut.L         698.907     20.335   34.369  < 2e-16 ***
cut.Q        -327.686     17.911  -18.295  < 2e-16 ***
cut.C         180.565     15.557   11.607  < 2e-16 ***
cut^4          -1.207     12.458   -0.097    0.923    
clarity.L    4217.535     30.831  136.794  < 2e-16 ***
clarity.Q   -1832.406     28.827  -63.565  < 2e-16 ***
clarity.C     923.273     24.679   37.411  < 2e-16 ***
clarity^4    -361.995     19.739  -18.339  < 2e-16 ***
clarity^5     216.616     16.109   13.447  < 2e-16 ***
clarity^6       2.105     14.037    0.150    0.881    
clarity^7     110.340     12.383    8.910  < 2e-16 ***
color.L     -1910.288     17.712 -107.853  < 2e-16 ***
color.Q      -627.954     16.121  -38.952  < 2e-16 ***
color.C      -171.960     15.070  -11.410  < 2e-16 ***
color^4        21.678     13.840    1.566    0.117    
color^5       -85.943     13.076   -6.572 5.00e-11 ***
color^6       -49.986     11.889   -4.205 2.62e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1157 on 53921 degrees of freedom
Multiple R-squared:  0.9159,    Adjusted R-squared:  0.9159 
F-statistic: 3.264e+04 on 18 and 53921 DF,  p-value: < 2.2e-16
```

The package `coefplot` contains the function `coefplot` that allows the visualization of the results of the algorithm. If we write

```
require(coefplot)
coefplat(priceRgr)
```

this will result in the plot shown in Figure 5.3.

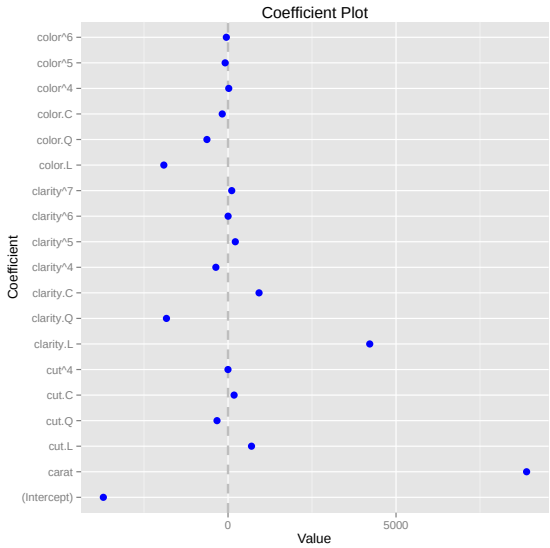

Fig. 5.3. Coefficient plot for price regression

6

Partitional Clustering

6.1 Introduction

Clustering is the process of grouping together objects that are similar. The groups formed by clustering are referred to as *clusters*.

Clustering is possible when we have a similarity measure that apply to pairs of objects, $s : S \times S \longrightarrow [0, 1]$, or a dissimilarity measure $d : S \times S \longrightarrow \mathbb{R}_{\geqslant 0}$ for objects.

The function s is a similarity if

- $s(x, x) = 1$ for $x \in S$;
- $s(x, y) = s(y, x)$ for $x, y \in S$.

The function $d : S \times S \longrightarrow \mathbb{R}_{\geqslant 0}$ is a dissimilarity if Axioms for dissimilarity:

- $d(x, x) = 0$ for $x \in S$;
- $d(x, y) = d(y, x)$ for $x, y \in S$.

If, in addition, $d(x, y) = 0$ implies $x = y$ for $x, y \in S$, we say that d is a *definite dissimilarity*.

There are several points of view for examining clustering techniques.

Clustering may or may not be *exclusive*, where an exclusive clustering technique yields clusters that are disjoint, while a nonexclusive technique produces overlapping clusters. From an algebraic point of view, an exclusive clustering algorithm generates a partition $\kappa = \{C_1, \ldots, C_k\}$ of the set of objects whose blocks $C_1, \ldots, C_k$ are referred to as *clusters*.

Clustering may be *intrinsic* or *extrinsic*. Intrinsic clustering is an unsupervised activity that is based only on the dissimilarities between the objects to be clustered. Most clustering algorithms fall into this category. Extrinsic clustering relies on information provided by an external source that prescribes, for example, which objects should be clustered together and which should not.

Finally, clustering may be *hierarchical* or *partitional*.

Partitional clustering creates a partition of the set of objects whose blocks are the clusters such that objects in a cluster are more similar to each other than to objects that belong to different clusters. A typical representative algorithm is the k-means algorithm and its many extensions.

In hierarchical clustering algorithms, a sequence of partitions is constructed. In *hierarchical agglomerative algorithms* this sequence is increasing and it begins with the least partition of the set of objects whose blocks consist of single objects; as the clustering progresses, certain clusters are fused together. As a result, an agglomerative clustering is a chain of partitions on the set of objects that begins with the least partition α_S of the set of objects S and ends with the largest partition ω_S. In a *hierarchical divisive algorithm*, the sequence of partitions is decreasing. Its first member is the one-block partition ω_S, and each partitions is built by subdividing the blocks of the previous partition.

Our presentation is organized around the last dichotomy. We start with partitional algorithms and continue with a class of hierarchical agglomerative algorithms. We conclude with an evaluation of clustering quality.

Clustering can be regarded as a special type of classification, where the clusters serve as classes of objects. It is a widely used data mining activity with multiple applications in a variety of scientific activities ranging from biology and astronomy to economics and sociology.

6.2 The *k*-Means Algorithm

The k-means algorithm is a partitional algorithm that requires the specification of the number of clusters k as an input. The set of objects to be clustered $\{\boldsymbol{o}^1, \ldots, \boldsymbol{o}^n\}$ is a subset of $\mathbb{R}^m$. Due to its simplicity and its many implementations it is a very popular algorithm despite this requirement.

When data to be clustered are numerical (that is, when $S \subseteq \mathbb{R}^n$), we can define the *centroid* of a nonempty subset U of S as:

$$\boldsymbol{c}_U = \frac{1}{|U|} \sum \{\boldsymbol{o} | \boldsymbol{o} \in U\}.$$

If $\pi = \{U_1, \ldots, U_m\}$ is a partition of S, then the *sum of the squared errors* of π is the number

$$sse(\pi) = \sum_{i=1}^{m} \sum \{d^2(\boldsymbol{o}, \boldsymbol{c}_{U_i}) | \boldsymbol{o} \in U_i\}, \tag{6.1}$$

where d is the Euclidean distance in $\mathbb{R}^n$.

The k-means algorithm begins with a randomly chosen collection of k centroids $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^k$ in $\mathbb{R}^m$. An initial partition of the set S of objects is computed by assigning each object $\boldsymbol{o}^i$ to its closest centroid $\boldsymbol{c}^j$. Let U_j be the set of points assigned to the centroid $\boldsymbol{c}^j$.

The assignments of objects to centroids are expressed by a matrix (b_{ij}), where

$$b_{ij} = \begin{cases} 1 & \text{if } \boldsymbol{o}^i \in U_j, \\ 0 & \text{otherwise.} \end{cases}$$

Since each object is assigned to exactly one cluster, we have $\sum_{j=1}^{k} b_{ij} = 1$. Also, $\sum_{i=1}^{n} b_{ij}$ equals the number of objects assigned to the centroid $\boldsymbol{c}^j$.

After these assignments, expressed by the matrix (b_{ij}), the centroids $\boldsymbol{c}^j$ must be re-computed using the formula:

$$\boldsymbol{c}^j = \frac{\sum_{i=1}^{n} b_{ij}\boldsymbol{o}^i}{\sum_{i=1}^{n} b_{ij}} \tag{6.2}$$

for $1 \leqslant j \leqslant k$.

The sum of squared errors of a partition $\pi = \{U_1, \ldots, U_k\}$ of a set of objects S was defined in Equality (6.1) as

$$sse(\pi) = \sum_{j=1}^{k} \sum_{\boldsymbol{o} \in U_j} d^2(\boldsymbol{o}, \boldsymbol{c}^j),$$

where $\boldsymbol{c}^j$ is the centroid of U_j for $1 \leqslant j \leqslant k$. The error of such an assignment is the sum of squared errors of the partition $\pi = \{U_1, \ldots, U_k\}$ defined as

$$\begin{aligned} sse(\pi) &= \sum_{i=1}^{n} \sum_{j=1}^{k} b_{ij} ||\boldsymbol{o}^i - \boldsymbol{c}^j||^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} b_{ij} \sum_{p=1}^{m} \left(o_p^i - c_p^j\right)^2 . \end{aligned}$$

The choice of the centroids is justified by the goal of obtaining local minima of the sum of squared errors of the clusterings.

The mk necessary conditions for a local minimum of $sse(\pi)$ are

$$\frac{\partial sse(\pi)}{\partial c_p^j} = \sum_{i=1}^{n} b_{ij} \left(-2(o_p^i - c_p^j)\right) = 0,$$

for $1 \leqslant p \leqslant m$ and $1 \leqslant j \leqslant k$, which can be written as

$$\sum_{i=1}^{n} b_{ij} o_p^i = \sum_{i=1}^{n} b_{ij} c_p^j = c_p^j \sum_{i=1}^{n} b_{ij},$$

or as

$$c_p^j = \frac{\sum_{i=1}^{n} b_{ij} o_p^i}{\sum_{i=1}^{n} b_{ij}}$$

for $1 \leqslant p \leqslant m$. In vectorial form, these conditions amount to

$$\boldsymbol{c}^j = \frac{\sum_{i=1}^{n} b_{ij}\boldsymbol{o}^i}{\sum_{i=1}^{n} b_{ij}},$$

which is exactly the formula (6.2) that is used to update the centroids.

Since we have new centroids, objects must be reassigned, which means that the values of b_{ij} must be recomputed, which, in turn, affects the values of the centroids, etc.

The halting criterion of the algorithm depends on particular implementations and may involve

(i) performing a certain number of iterations;
(ii) lowering the sum of squared errors $sse(\pi)$ is below a certain limit;
(iii) the current partition coinciding with the previous partition.

Algorithm 6.2.1: The k-means Algorithm

Data: the set of objects to be clustered $S = \{\boldsymbol{o}^1, \ldots, \boldsymbol{o}^n\}$ and the number of clusters k
Result: collection of k clusters

```
extract a randomly chosen collection of k vectors c_1, ..., c_k in R^n;
assign each object o^i to the closest centroid c^j;
let π = {U_1, ..., U_k} be the partition defined by c^1, ..., c^k;
recompute the centroids of the clusters U_1, ..., U_k;
while halting criterion is not met do
    compute the new value of the partition π using the current centroids;
    recompute the centroids of the blocks of π;
end
```

The popularity of the k-means algorithm stems from its simplicity and its low time complexity $O(kn\ell)$, where n is the number of objects to be clustered and ℓ is the number of iterations that the algorithm is performing.

There are several variants of the k-means algorithm and some of them are implemented in **R**, as we shall see; these variants are designated as: "Forgy" (or "Lloyd), "MacQueen", and "Hartigan-Wong, and, unless specified otherwise by the parameter algorithm, the default is "Hartigan-Wong".

In the "Forgy" variant k instances are randomly chosen and used as the initial centroids. This approach takes advantage of the fact that the random selection increases the likelihood to choose a point near a cluster centre because this is where the highest density of points is located. However, it may happen that we will choose two initial centroids near the centre of the same cluster.

Later, McQueen proposed that after k instances are chosen at random, one instance is assigned at a time to the nearest cluster centre. After each instance is assigned, the k-means algorithm is run before the next instance is assigned to a cluster. Since several iterations of the k- means algorithm are needed after each instance is assigned, the algorith which is expensive when the number of objects is high.

The Hartigan-Wong variant of the k-means algorithm redistributes objects to clusters based on the effect of such a reassignment on the objective function.

If $sse(\pi)$ decreases, the object is moved and the two centroids of the affected clusters are recomputed.

The function `kmeans` of **R** performs the k-means clustering on a data matrix. The standard usage of this function is

```
kmeans(D, centers, iter.max = 10, nstart = 1, algorithm,trace)
```

and its arguments are:

- `D`: a matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns);
- `centers`: either the number of clusters, say k, or a set of initial (distinct) cluster centroids. In the first case, a random set of (distinct) rows in D is chosen as the set of initial centroids;
- `iter.max`: the maximum number of iterations allowed;
- `nstart`: if `centers` is a number, this parameter indicates the number of random sets;
- `algorithm`: a string that indicates the variant of the algorithm, as discussed previously;
- `trace`: logical or integer number, currently only used in the default method (`"Hartigan-Wong"`): if positive (or true), tracing information on the progress of the algorithm is produced; higher values may produce more tracing information.

The function `kmeans` returns an object which has a print and a fitted method. It is a list with at least the following components:

- `cluster`: a vector of integers (from 1 to k) indicating the cluster to which each point is allocated;
- `centers`: a matrix of cluster centres;
- `totss`: the total sum of squares;
- `withinss`: vector of within-cluster sum of squares, one component per cluster;
- `tot.withinss`: total within-cluster sum of squares, that is, the sum(withinss);
- `betweenss`: the between-cluster sum of squares;
- `size`: the number of points in each cluster;
- `iter`: the number of (outer) iterations;
- `ifault`: an integer indicator of a possible algorithm problem.

Example 25. We use the function `setofpoints2` defined by

```
setofpoints2 <- function(n,center,stdev){
                        return(cbind(rnorm(n,center[1],stdev[1]),
                                     rnorm(n,center[2],stdev[2])))
}
```

to generate n points in $\mathbb{R}^2$ normally distributed around the vector `center` and having the standard deviations specified by the vector `stdev`. We begin by producing three sets of points `A`, `B` and `D`, and, then by joining these sets into the set `D` and naming the columns of this matrix as `("x")` and `("y")`:

```
A <- setofpoints2(40,c(2,4),c(0.2,0.3))
B <- setofpoints2(30,c(3,2.5),c(0.3,0.3))
C <- setofpoints2(45,c(4,4),c(0.2,0.4))
D <- rbind(A,B,C)
colnames(D) <- c("x","y")
```

The plot of the set `D` is shown in Figure 6.1.

The kmeans cluster produces the object `sp` in

```
sp <-kmeans(D,3)
```

that is plotted in Figure 6.2 using `plot(D,col=sp$cluster)`.

6.3 The PAM Algorithm

Another algorithm, named PAM (an acronym of "Partition Around Medoids") developed by Kaufman and Rousseeuw, also requires as an input parameter the number k of clusters to be extracted.

The k clusters are determined based on a representative object from each cluster, called the *medoid* of the cluster. The medoid of a cluster is one of the objects that have a most central position in the cluster.

PAM begins with a set of objects S, where $|S| = n$, a dissimilarity $n \times n$ matrix D, and a prescribed number of clusters k. The d_{ij} entry of the matrix D is the dissimilarity $d(o_i, o_j)$ between the objects o_i and o_j.

The algorithm has two phases:

(i) the *building phase*, and
(ii) the *swapping phase*.

The building phase aims to construct a set L of selected objects, $L \subseteq S$. The set of remaining objects is denoted by R; clearly, $R = S - L$. To determine the most centrally located object we compute $Q_i = \sum_{j=1}^{n} d_{ij}$ for $1 \leqslant i \leqslant n$. The most central object o_q is determined by $q = \arg\min_i Q_i$. The set L is initialized as $L = \{o_q\}$.

Suppose now that we have constructed a set L of selected objects and $|L| < k$. We need to add a new selected object to the set L. To do this, we need to examine all objects that have not been included in L so far, that is, all objects in R. The selection is determined by a merit function $M : R \longrightarrow \mathbb{N}$.

To compute the merit $M(o)$ of an object $o \in R$, we scan all objects in R distinct from o. Let $o' \in R - \{o\}$ be such an object. If $d(o, o') < d(L, o')$, then adding o to L could benefit the clustering (from the point of view of o') because $d(L, o')$ will diminish. The potential benefit is $d(o', L) - d(o, o')$. Of course, if $d(o, o') \geqslant d(L, o')$, no such benefit exists (from the point of view of o'). Thus, we compute the merit of o as

$$M(o) = \sum_{o' \in R - \{o\}} \max\{D(L, o') - d(o, o'), 0\}.$$

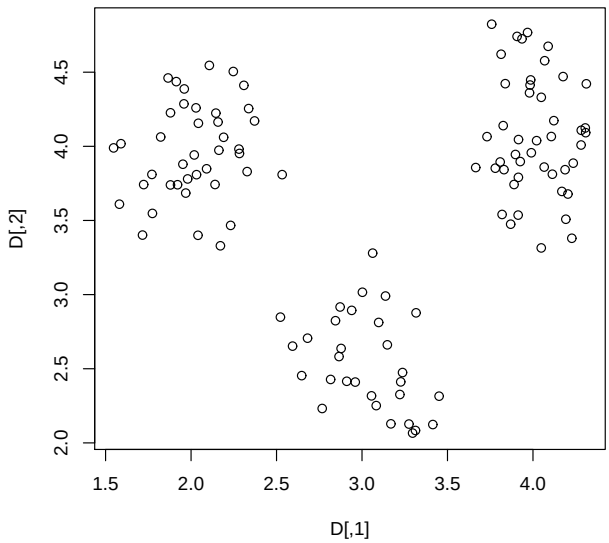

Fig. 6.1. Set of Points to be Clustered

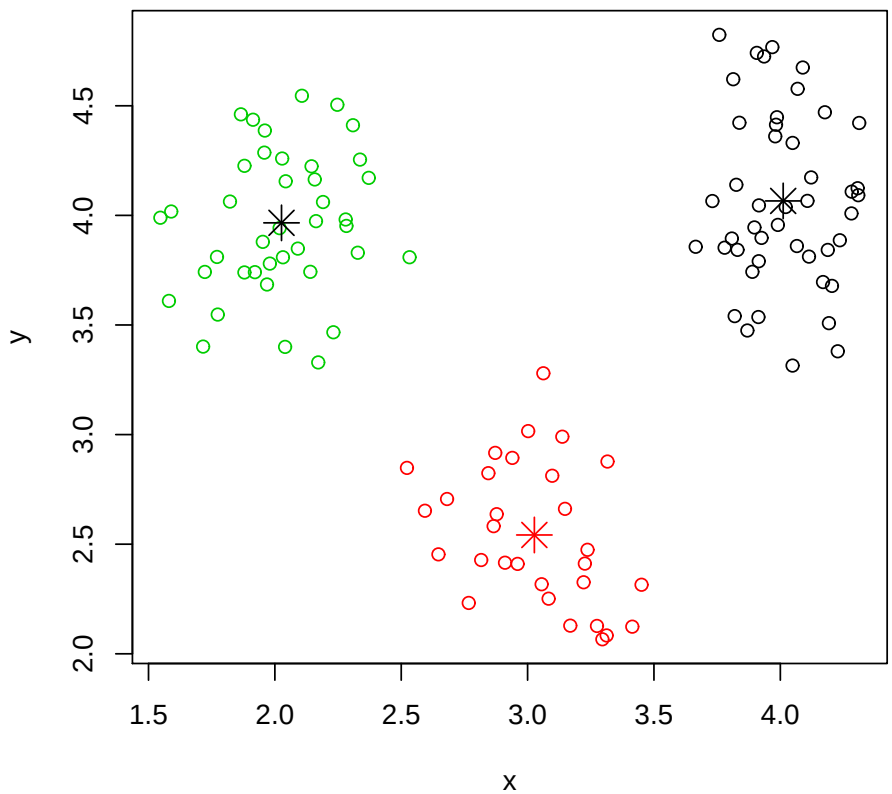

Fig. 6.2. Plot of clustered objects

We add to L the unselected object o that has the largest merit value. The building phase halts when $|L| = k$.

The objects in set L are the potential medoids of the k clusters that we seek to build.

In the *swapping phase* swapping objects and existing medoids is considered. A cost of a swap is defined with the intention of penalizing swaps that diminish the centrality of the medoids in the clusters. Swapping continues as long as useful swaps (that is, swaps with negative costs) can be found.

Let o_i be a selected object, $o_i \in L$, and let o_h be an unselected object, $o_h \in R = S - L$. The cost $C(o_i, o_h)$ of swapping o_i and o_h is determined by the contribution of each unselected object o_j.

Let o_j be an arbitrary unselected object and let $d(L, o_j)$ be the least dissimilarity between o_j and an object in L. Also, let $e(o_j)$ be the dissimilarity between o_j and the second most similar object in L.

The contribution c_{ihj} of o_j to the cost of the swap between o_i and o_h is defined as follows:

1. If $d(o_i, o_j)$ and $d(o_h, o_j)$ are greater than $d(o, o_j)$ for any $o \in L - \{o_i\}$, then $c_{ihj} = 0$.
2. If $d(o_i, o_j) = d(L, o_j)$, then two cases must be considered depending $e(o_j)$:
   a) If $d(o_h, o_j) < e(o_j)$, then $c_{ihj} = d(o_h, o_j) - d(L, o_j)$.
   b) If $d(o_h, o_j) \geqslant e(o_j)$, then $c_{ihj} = e(o_j) - d(L, o_j)$.
3. If $d(o_i, o_j) > d(L, o_j) > d(o_h, o_j)$ (that is, o_j is more distant from o_i than from at least one other selected object and o_j is closer to o_h than to any selected object), then $c_{ihj} = d(o_h, o_j) - d(L, o_j)$.

The cost of the swap is $C(o_i, o_h) = \sum_{o_j \in R} c_{ihj}$. The pair that minimizes $C(o_i, o_j)$ is selected. If $C(o_i, o_j) < 0$, then the swap is carried out. All potential swaps are considered.

The algorithm halts when no useful swap exists; that is, no swap with negative cost can be found.

PAM is more robust than k-clustering because it minimizes the sum of the dissimilarities instead of the sum of the squared errors.

The pseudocode of the algorithm is given in Algorithm 6.3.1.

Algorithm 6.3.1: The PAM algorithms

Data: a set of objects S, where $|S| = n$, a dissimilarity $n \times n$ matrix D, and a prescribed number of clusters k
Result: a k-clustering of S

```
construct the set L of k medoids;
repeat
    compute the costs C(o_i, o_h) for o_i ∈ L and o_h ∈ R;
    select the pair (o_i, o_h) that corresponds to the minimum m = C(o_i, o_h);
until (m > 0);
```

Note that inside the loop **repeat** $\cdots$ **until** there are $l(n-l)$ pairs of objects to be examined, and for each pair we need to involve $n-l$ non-selected objects. Thus, one execution of the loop requires $O(l(n-l)^2)$, and the total execution may require up to $O\left(\sum_{l=1}^{n-l} l(n-l)^2\right)$, which is $O(n^4)$. Thus, the usefulness of PAM is limited to rather small data set (no more than a few hundred objects).

The function `pam`, a component of the package `clust` implements the algorithm discussed above.

Example 26. To apply the algorithm to the data set `D` previously computed we write:

```
pamx <- pam(D, 3)
```

The summary of the object `pamx` returned by `summary(pamx)` is

```
Medoids:
     ID        x        y
[1,] 39 2.018645 3.941830
[2,] 48 2.960442 2.410117
[3,] 85 4.021261 4.038188
Clustering vector:
  [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
 [38] 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 2 2 2 2 2 3 3 3 3
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3
Objective function:
    build      swap
0.4756183 0.3719818

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
 [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Clusters can be visualized using the function `clusplot` which result in the representation shown in Figure 6.3.

6.4 Evaluation of Clusterings

The *silhouette method* is an unsupervised method for evaluation of clusterings that computes certain coefficients for each object. The set of these coefficients allows an evaluation of the quality of the clustering.

Let $S = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n\}$ be a collection of objects, $d : S \times S \longrightarrow \mathbb{R}_{\geq 0}$ a dissimilarity on S, and let $\{C_1, \ldots, C_k\}$ be a clustering of S.

Suppose that $\boldsymbol{o}_i \in C_\ell$. The *average dissimilarity* of $\boldsymbol{o}_i$ is given by

$$a(\boldsymbol{o}_i) = \frac{\sum\{d(\boldsymbol{o}_i, \boldsymbol{u}) \mid u \in C_\ell - \{\boldsymbol{o}_i\}\}}{|C_\ell|},$$

that is, the average dissimilarity between $\boldsymbol{o}_i$ to all other objects of C_ℓ, the cluster to which $\boldsymbol{o}_i$ is assigned.

For $\boldsymbol{o}_i$ and a cluster $C \neq C_\ell$ let

$$d(\boldsymbol{o}_i, C) = \frac{\sum\{d(\boldsymbol{o}_i, u) \mid f(u) = C\}}{|C|},$$

be the average dissimilarity between $\boldsymbol{o}_i$ and the objects of the cluster C.

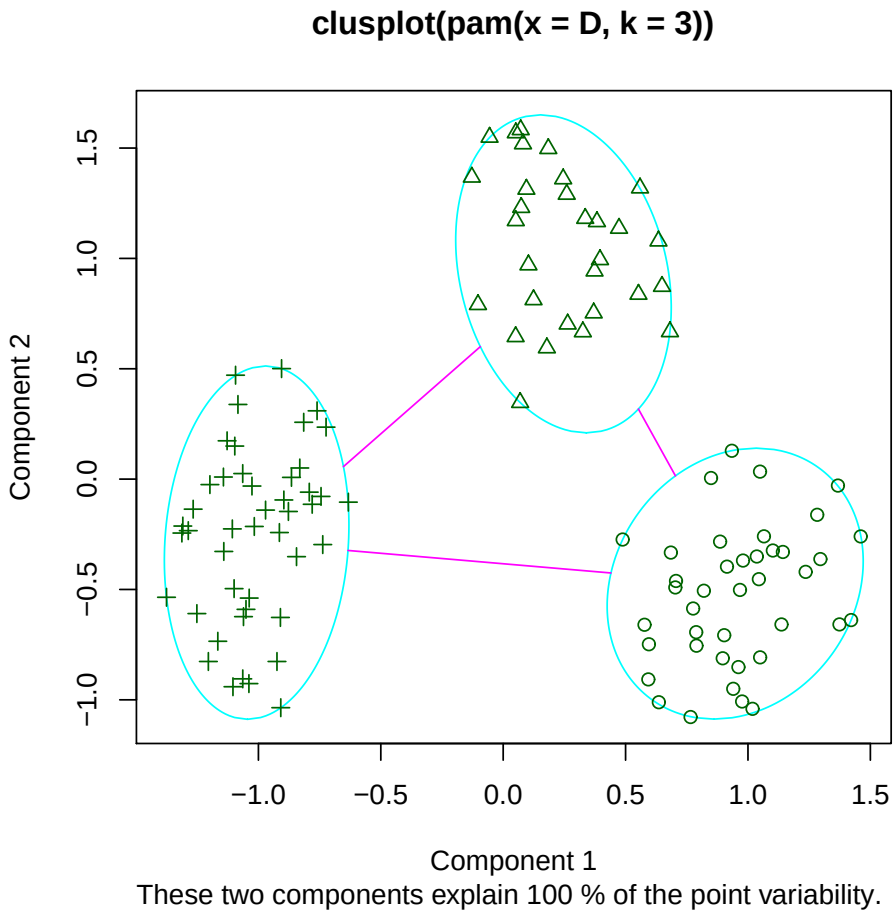

Fig. 6.3. Plot of clustered objects

Definition 5. *Let $\{C_1, \ldots, C_k\}$ be a clustering. A neighbor of $\boldsymbol{o}_i$ is a cluster $C \neq C_\ell$ for which $d(\boldsymbol{o}_i, C)$ is minimal.*

In other words, a neighbor of an object $\boldsymbol{o}_i$ is "the second best choice" for a cluster for $\boldsymbol{o}_i$. Let $b : S \longrightarrow \mathbb{R}_{\geq 0}$ be the function defined by

$$b(\boldsymbol{o}_i) = \min\{d(\boldsymbol{o}_i, C) \mid C \neq C_\ell\}.$$

Definition 6. *The silhouette of the object $\boldsymbol{o}_i$ for which $|C_\ell| \geq 2$ is the number $sil(\boldsymbol{o}_i)$ given by*

$$sil(\boldsymbol{o}_i) = \begin{cases} 1 - \frac{a(\boldsymbol{o}_i)}{b(\boldsymbol{o}_i)} & \textit{if } a(\boldsymbol{o}_i) < b(\boldsymbol{o}_i) \\ 0 & \textit{if } a(\boldsymbol{o}_i) = b(\boldsymbol{o}_i) \\ \frac{b(\boldsymbol{o}_i)}{a(\boldsymbol{o}_i)} - 1 & \textit{if } a(\boldsymbol{o}_i) > b(\boldsymbol{o}_i). \end{cases}$$

Equivalently, we have

$$sil(\boldsymbol{o}_i) = \frac{b(\boldsymbol{o}_i) - a(\boldsymbol{o}_i)}{\max\{a(\boldsymbol{o}_i), b(\boldsymbol{o}_i)\}}$$

for $\boldsymbol{o}_i \in O$.

Observe that $-1 \leq sil(\boldsymbol{o}_i) \leq 1$. When $sil(\boldsymbol{o}_i)$ is close to 1, this means that $a(\boldsymbol{o}_i)$ is much smaller than $b(\boldsymbol{o}_i)$ and we may conclude that $\boldsymbol{o}_i$ is well-classified. When $sil(\boldsymbol{o}_i)$ is near 0, it is not clear which is the best cluster for $\boldsymbol{o}_i$. Finally, if $sil(\boldsymbol{o}_i)$ is close to -1, the average distance from u to its neighbor(s) is much smaller than the average distance between $\boldsymbol{o}_i$ and other objects that belong to the same cluster $f(\boldsymbol{o}_i)$. In this case, it is clear that $\boldsymbol{o}_i$ is poorly classified.

Definition 7. *The average silhouette width of a cluster* C *is*

$$sil(C) = \frac{\sum\{sil(u) \mid u \in C\}}{|C|}.$$

The average silhouette width of a clustering κ *is*

$$sil(\kappa) = \frac{\sum\{sil(u) \mid u \in O\}}{|O|}.$$

The silhouette of a clustering can be used for determining the "optimal" number of clusters. If the average silhouette of the clustering is above 0.7, we have a strong clustering.

Example 27. The sihouette plot of the `pamx` clustring discussed in Example 26 is shown in Figure 6.4. and is obtained with the command

```
sil <- silhouette(pamx)
plot(sil)
```

The function silhouette is a part of the package `clust`.

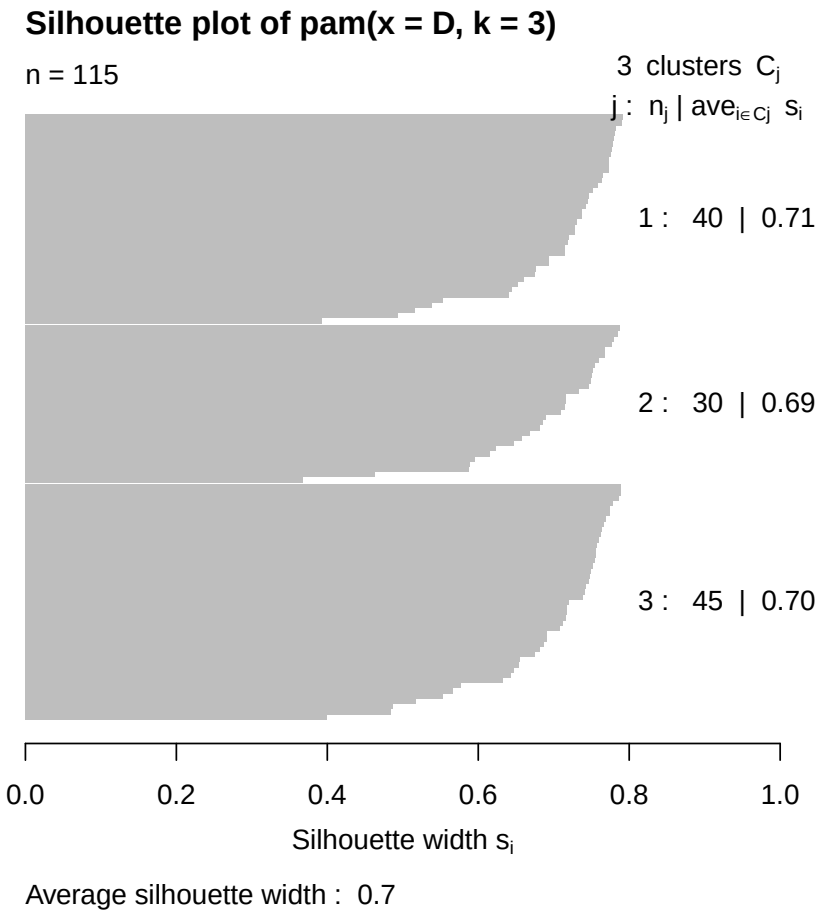

Fig. 6.4. Silhoutee of the PAM clustering

7

Hierarchical Clustering

7.1 Introduction

7.2 Ultrametrics and Ultrametric Spaces

An ultrametric on a set S is a mapping $d : S^2 \longrightarrow \mathbb{R}_{\geqslant 0}$ that has the following properties:

(i) $d(x, y) = 0$ if and only if $x = y$ for $x, y \in S$;
(ii) $d(x, y) = d(y, x)$ for $x, y \in S$;
(iii) $d(x, y) \leqslant \max\{d(x, z), d(z, y)\}$ for $x, y, z \in S$.

If property (i) is replaced by the weaker requirement that $d(x, x) = 0$ for $x \in S$, then d is a *quasi-ultrametric* on S.

Example 28. Let $\pi = \{B, C\}$ be a two-block partition of a nonempty set S. Define the mapping $d_\pi : S^2 \longrightarrow \mathbb{R}_{\geqslant 0}$ by

$$d_\pi(x, y) = \begin{cases} 0 & \text{if } \{x, y\} \subseteq B \text{ or } \{x, y\} \subseteq C \\ 1 & \text{otherwise,} \end{cases}$$

for $x, y \in S$. We claim that d_π is a quasi-ultrametric. Indeed, it is clear that $d_\pi(x, x) = 0$ and $d_\pi(x, y) = d_\pi(y, x)$ for $x, y \in S$. Now let x, y, z be three arbitrary elements in S. If $d_\pi(x, y) = 1$, then x and y belong to two distinct blocks of the partition π, say to B and C, respectively. If $z \in B$, then $d_\pi(x, z) = 0$ and $d_\pi(z, y) = 1$; similarly, if $z \in C$, then $d_\pi(x, z) = 1$ and $d_\pi(z, y) = 0$. In either case, the ultrametric inequality is satisfied.

Example 29. Let A be a finite set and let $A^{\mathbb{N}}$ be the set of functions of the form $f : \mathbb{N} \longrightarrow A$. For $f, g : \mathbb{N} \longrightarrow A$ define $\delta(f, g) = \min\{n \in \mathbb{N} \mid f(n) \neq g(n)\}$ and $d(f, g) = 2^{-\delta(f,g)}$.

We claim that d is an ultrametric. Let $f, g, h : \mathbb{N} \longrightarrow A$ and let $\delta(f, g) = p$ and $\delta(g, h) = q$. Suppose that $p \leqslant q$. In this case, it follows that $\delta(f, h) = p$. Thus, $d(f, g) = 2^{-p} = d(f, h) \geqslant d(g, h)$ and the ultrametric inequality

$d(f,g) \leqslant \max\{d(f,h), d(g,h)\}$ is satisfied. The remaining cases are left to the reader.

Theorem 11. *Let $a_0, a_1, a_2 \in \mathbb{R}$ be three numbers. If $a_i \leqslant \max\{a_j, a_k\}$ for every permutation (i,j,k) of the set $\{0,1,2\}$, then two of the numbers are equal and the third is not larger than the two others.*

Proof. Suppose that a_i is the least of the numbers a_0, a_1, a_2 and a_j, a_k are the remaining numbers. Since $a_j \leqslant \max\{a_i, a_k\} = a_k$ and $a_k \leqslant \max\{a_i, a_j\} = a_j$, it follows that $a_j = a_k \geqslant a_i$.

Triangles in ultrametric spaces have an interesting property that is given next.

Corollary 2. *Let (S,d) be an ultrametric space. For every $x, y, z \in S$, two of the numbers $d(x,y), d(x,z), d(y,z)$ are equal and the third is not larger than the two other equal numbers.*

Proof. Since d satisfies the ultrametric inequality, the statement follows immediately from Theorem 11.

Theorem 12. *Let $B(x,r)$ be a closed sphere in the ultrametric space (S,d). If $z \in B(x,d)$, then $B(x,r) = B(z,r)$. In other words, in an ultrametric space, a closed sphere has all its points as centers.*

Proof. Suppose that $z \in B(x,r)$, so $d(x,z) \leqslant r$. Let $y \in B(z,r)$. Since $d(y,x) \leqslant \max\{d(y,z), d(z,x)\} \leqslant r$, we have $y \in B(x,r)$. Conversely, if $y \in B(x,r)$, we have $d(y,z) \leqslant \max\{d(y,x), d(x,z)\} \leqslant r$, hence $y \in B(z,r)$.

Both closed and open spheres in ultrametric spaces are clopen sets as we show next.

Theorem 13. *If d is an ultrametric on S, then any closed sphere $B(t,r)$ and any open sphere $C(t,r)$ are clopen sets in the topological ultrametric space $(S, \mathcal{O}_d)$.*

Proof. We already know that $B(t,r)$ is closed. To prove that this set is also open if d is an ultrametric, let $s \in B(t,r)$. By Theorem 12 s is a center of the sphere. Therefore, $C\left(s, \frac{r}{2}\right) \subseteq B(t,r)$, so $B(t,r)$ is open. We leave the proof that $C(t,r)$ is also closed to the reader.

By Theorem **??**, the border of a closed sphere or of an open sphere in an ultrametric space is empty.

Theorem 14. *Let (S,d) be an ultrametric space, $x, y \in S$, and let $S(x,y) \subseteq \mathbf{Seq}(S)$ be the set of sequences that start with x and end with y. We have $d(x,y) = \min\{amp_d(\boldsymbol{s}) \mid \boldsymbol{s} \in S(x,y)\}$.*

Proof. Since d is an ultrametric, we have $d(x,y) \leqslant amp_d(\boldsymbol{s})$ for any nonnull sequence $\boldsymbol{s} = (s_1, \ldots, s_n)$ such that $s_1 = x$ and $s_n = y$. Therefore,

$$d(x,y) \leqslant \min\{amp_d(\boldsymbol{s}) \mid \boldsymbol{s} \in S(x,y)\}.$$

The equality of the theorem follows from the fact that $(x,y) \in S(x,y)$.

Theorem 15. *If two closed spheres $B(x,r)$ and $B(y,r')$ of an ultrametric space have a point in common, then one of the closed spheres is included in the other.*

Proof. The statement follows directly from Theorem 12.

Theorem 12 implies that the entire space S equals the closed sphere $B(x, diam_{S,d})$ for any point $x \in S$.

The next statement gives a method of constructing ultrametrics starting from chains of equivalence relations.

Theorem 16. *Let S be a finite set and let $d : S \times S \longrightarrow \mathbb{R}_{\geqslant 0}$ be a function whose range is $\mathsf{range}(d) = \{r_1, \ldots, r_m\}$, where $r_1 = 0$ such that $d(x,y) = 0$ if and only if $x = y$. Define the relations $\eta_{r_i} = \{(x,y) \in S \times S \mid d(x,y) \leqslant r_i\}$ for $1 \leqslant i \leqslant m$.*

The function d is an ultrametric on S if and only if the sequence of relations $\eta_{r_1}, \ldots, \eta_{r_m}$ is an increasing chain of equivalences on S such that $\eta_{r_1} = \iota_S$ and $\eta_{r_m} = \theta_S$.

Proof. Suppose that d is an ultrametric on S. We have $(x,x) \in \eta_{r_i}$ because $d(x,x) = 0$, so all relations η_{r_i} are reflexive. Also, it is clear that the symmetry of d implies $(x,y) \in \eta_{r_i}$ if and only if $(y,x) \in \eta_{r_i}$, so these relations are symmetric.

The ultrametric inequality is essential for proving the transitivity of the relations η_{r_i}. If $(x,y),(y,z) \in \eta_{r_i}$, then $d(x,y) \leqslant r_i$ and $d(y,z) \leqslant r_i$, which implies $d(x,z) \leqslant \max\{d(x,y), d(y,z)\} \leqslant r_i$. Thus, $(x,z) \in \eta_{r_i}$, which shows that every relation η_{r_i} is transitive and therefore an equivalence.

It is straightforward to see that $\eta_{r_1} \leqslant \eta_{r_2} \leqslant \cdots \leqslant \eta_{r_m}$; that is, this sequence of relations is indeed a chain of equivalences.

Conversely, suppose that $\eta_{r_1}, \ldots, \eta_{r_m}$ is an increasing sequence of equivalences on S such that $\eta_{r_1} = \iota_S$ and $\eta_{r_m} = \theta_S$, where $\eta_{r_i} = \{(x,y) \in S \times S \mid d(x,y) \leqslant r_i\}$ for $1 \leqslant i \leqslant m$ and $r_1 = 0$.

Note that $d(x,y) = 0$ is equivalent to $(x,y) \in \eta_{r_1} = \iota_S$, that is, to $x = y$.

We claim that

$$d(x,y) = \min\{r \mid (x,y) \in \eta_r\}. \tag{7.1}$$

Indeed, since $\eta_{r_m} = \theta_S$, it is clear that there is an equivalence η_{r_i} such that $(x,y) \in \eta_{r_i}$. If $(x,y) \in \eta_{r_i}$, the definition of η_{r_i} implies $d(x,y) \leqslant r_i$, so $d(x,y) \leqslant \min\{r \mid (x,y) \in \eta_r\}$. This inequality can be easily seen to become an equality since $(x,y) \in \eta_{d(x,y)}$. This implies immediately that d is symmetric.

To prove that d satisfies the ultrametric inequality, let x, y, z be three members of the set S. Let $p = \max\{d(x,z), d(z,y)\}$. Since $(x,z) \in \eta_{d(x,z)} \subseteq \eta_p$ and $(z,y) \in \eta_{d(z,y)} \subseteq \eta_p$, it follows that $(x,y) \in \eta_p$, due to the transitivity of the equivalence η_p. Thus, $d(x,y) \leqslant p = \max\{d(x,z), d(z,y)\}$, which proves the triangular inequality for d.

Of course, Theorem 16 can be formulated in terms of partitions.

Theorem 17. *Let S be a finite set and let $d : S \times S \longrightarrow \mathbb{R}_{\geqslant 0}$ be a function whose range is* $\mathsf{range}(d) = \{r_1, \ldots, r_m\}$*, where $r_1 = 0$ such that $d(x,y) = 0$ if and only if $x = y$. For $u \in S$ and $r \in \mathbb{R}_{\geqslant 0}$, define the set $D_{u,r} = \{x \in S \mid d(u,x) \leqslant r\}$ and let $\pi_{r_i} = \{D(u, r_i) \mid u \in S\}$ for $1 \leqslant i \leqslant m$.*

The function d is an ultrametric on S if and only if the sequence $\pi_{r_1}, \ldots, \pi_{r_m}$ is an increasing sequence of partitions on S such that $\pi_{r_1} = \alpha_S$ and $\pi_{r_m} = \omega_S$.

Proof. The argument is entirely similar to the proof of Theorem 16 and is omitted.

Hierarchies and Ultrametrics

Definition 8. *Let S be a set. A* hierarchy *on S is a collection of sets $\mathcal{H} \subseteq \mathcal{P}(S)$ that satisfies the following conditions:*
(i) *the members of $\mathcal{H}$ are nonempty sets;*
(ii) *$S \in \mathcal{H}$;*
(iii) *for every $x \in S$, we have $\{x\} \in \mathcal{H}$;*
(iv) *if $H, H' \in \mathcal{H}$ and $H \cap H' \neq \emptyset$, then we have either $H \subseteq H'$ or $H' \subseteq H$.*

A standard technique for constructing a hierarchy on a set S starts with a rooted tree $(\mathcal{T}, v_0)$ whose nodes are labeled by subsets of the set S. Let V be the set of vertices of the tree $\mathcal{T}$. The function $\mu : V \longrightarrow \mathcal{P}(S)$, which gives the label $\mu(v)$ of each node $v \in V$, is defined as follows:
(i) the tree $\mathcal{T}$ has $|S|$ leaves, and each leaf v is labeled by a distinct singleton $\mu(v) = \{x\}$ for $x \in S$;
(ii) if an interior vertex v of the tree has the descendants $v_1, v_2, \ldots, v_n$, then $\mu(v) = \bigcup_{i=1}^{n} \mu(v_i)$.

The set of labels $\mathcal{H}_{\mathcal{T}}$ of the rooted tree $(\mathcal{T}, v_0)$ forms a hierarchy on S. Indeed, note that each singleton $\{x\}$ is a label of a leaf. An easy argument by induction on the height of the tree shows that every vertex is labeled by the set of labels of the leaves that descend from that vertex. Therefore, the root v_0 of the tree is labeled by S.

Suppose that H, H' are labels of the nodes u, v of $\mathcal{T}$, respectively. If $H \cap H' \neq \emptyset$, then the vertices u, v have a common descendant. In a tree, this can take place only if u is a descendant of v or v is a descendant of u; that is, only if $H \subseteq H'$, or $H' \subseteq H$, respectively. This gives the desired conclusion.

Example 30. Let $S = \{s, t, u, v, w, x, y\}$ and let $\mathcal{T}$ be a tree whose vertices are labeled as shown in Figure 7.4. It is easy to verify that the family of subsets of S that label the nodes of $\mathcal{T}$,

$$\begin{aligned}\mathcal{H} = \{&\{s\}, \{t\}, \{u\}, \{v\}, \{w\}, \{x\}, \{y\},\\ &\{s, t, u\}, \{w, x\}, \{s, t, u, v\}, \{w, x, y\}, \{s, t, u, v, w, x, y\}\}\end{aligned}$$

is a hierarchy on the set S.

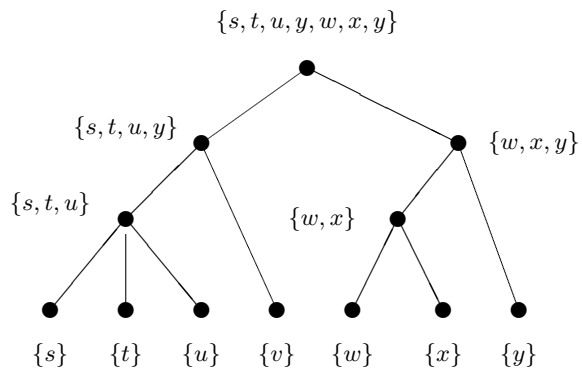

Fig. 7.1. Tree labeled by subsets of S.

Chains of partitions defined on a set generate hierarchies, as we show next.

Theorem 18. *Let S be a set and let $C = (\pi_1, \pi_2, \ldots, \pi_n)$ be an increasing chain of partitions $(PART(S), \leqslant)$ such that $\pi_1 = \alpha_S$ and $\pi_n = \omega_S$. Then, the collection $\mathcal{H}_C = \bigcup_{i=1}^{n} \pi_i$ that consists of the blocks of all partitions in the chain is a hierarchy on S.*

Proof. The blocks of any of the partitions are nonempty sets, so $\mathcal{H}_C$ satisfies the first condition of Definition 10.

We have $S \in \mathcal{H}_C$ because S is the unique block of $\pi_n = \omega_S$. Also, since all singletons $\{x\}$ are blocks of $\alpha_S = \pi_1$, it follows that $\mathcal{H}_C$ satisfies the second and the third conditions of Definition 10. Finally, let H and H' be two sets of $\mathcal{H}_C$ such that $H \cap H' \neq \emptyset$. Because of this condition, it is clear that these two sets cannot be blocks of the same partition. Thus, there exist two partitions π_i and π_j in the chain such that $H \in \pi_i$ and $H' \in \pi_j$. Suppose that $i < j$. Since every block of π_j is a union of blocks of π_i, H' is a union of blocks of π_i and $H \cap H' \neq \emptyset$ means that H is one of these blocks. Thus, $H \subseteq H'$. If $j > i$, we obtain the reverse inclusion. This allows us to conclude that $\mathcal{H}_C$ is indeed a hierarchy.

Theorem 25 can be stated in terms of chains of equivalences; we give the following alternative formulation for convenience.

Theorem 19. *Let S be a finite set and let $(\rho_1, \ldots, \rho_n)$ be a chain of equivalence relations on S such that $\rho_1 = \iota_S$ and $\rho_n = \theta_S$. Then, the collection of blocks of the equivalence relations ρ_r (that is, the set $\bigcup_{1 \leqslant r \leqslant n} S/\rho_r$) is a hierarchy on S.*

Proof. The proof is a mere restatement of the proof of Theorem 25.

Define the relation "$\prec$" on a hierarchy $\mathcal{H}$ on S by $H \prec K$ if $H, K \in \mathcal{H}$, $H \subset K$, and there is no set $L \in \mathcal{H}$ such that $H \subset L \subset K$.

Lemma 1. *Let $\mathcal{H}$ be a hierarchy on a finite set S and let $L \in \mathcal{H}$. The collection $\mathcal{P}_L = \{H \in \mathcal{H} \mid H \prec L\}$ is a partition of the set L.*

Proof. We claim that $L = \bigcup \mathcal{P}_L$. Indeed, it is clear that $\bigcup \mathcal{P}_L \subseteq L$.

Conversely, suppose that $z \in L$ but $z \notin \bigcup \mathcal{P}_L$. Since $\{z\} \in \mathcal{H}$ and there is no $K \in \mathcal{P}_L$ such that $z \in K$, it follows that $\{z\} \in \mathcal{P}_L$, which contradicts the assumption that $z \notin \bigcup \mathcal{P}_L$. This means that $L = \bigcup \mathcal{P}_L$.

Let $K_0, K_1 \in \mathcal{P}_L$ be two distinct sets. These sets are disjoint since otherwise we would have either $K_0 \subset K_1$ or $K_1 \subset K_0$, and this would contradict the definition of $\mathcal{P}_L$.

Theorem 20. *Let $\mathcal{H}$ be a hierarchy on a set S. The graph of the relation $\prec$ on $\mathcal{H}$ is a tree whose root is S; its leaves are the singletons $\{x\}$ for every $x \in S$.*

Proof. Since $\prec$ is an antisymmetric relation on $\mathcal{H}$, it is clear that the graph $(\mathcal{H}, \prec)$ is acyclic. Moreover, for each set $K \in \mathcal{H}$, there is a unique path that joins K to S, so the graph is indeed a rooted tree.

Definition 9. *Let $\mathcal{H}$ be a hierarchy on a set S. A* grading function *for $\mathcal{H}$ is a function $h : \mathcal{H} \longrightarrow \mathbb{R}$ that satisfies the following conditions:*
(i) *$h(\{x\}) = 0$ for every $x \in S$, and*
(ii) *if $H, K \in \mathcal{H}$ and $H \subset K$, then $h(H) < h(K)$.*

If h is a grading function for a hierarchy $\mathcal{H}$, the pair $(\mathcal{H}, h)$ is a graded hierarchy*.*

Example 31. For the hierarchy $\mathcal{H}$ defined in Example 35 on the set $S = \{s, t, u, v, w, x, y\}$, the function $h : \mathcal{H} \longrightarrow \mathbb{R}$ given by

$$\begin{aligned}
&h(\{s\}) = h(\{t\}) = h(\{u\}) = h(\{v\}) = h(\{w\}) = h(\{x\}) = h(\{y\}) = 0,\\
&h(\{s,t,u\}) = 3, h(\{w,x\}) = 4, h(\{s,t,u,v\}) = 5, h(\{w,x,y\}) = 6,\\
&h(\{s,t,u,v,w,x,y\}) = 7,
\end{aligned}$$

is a grading function and the pair $(\mathcal{H}, h)$ is a graded hierarchy on S.

Theorem 25 can be extended to graded hierarchies.

Theorem 21. *Let S be a finite set and let $C = (\pi_1, \pi_2, \ldots, \pi_n)$ be an increasing chain of partitions $(PART(S), \leqslant)$ such that $\pi_1 = \alpha_S$ and $\pi_n = \omega_S$.*

If $f : \{1, \ldots, n\} \longrightarrow \mathbb{R}_{\geqslant 0}$ is a function such that $f(1) = 0$, then the function $h : \mathcal{H}_C \longrightarrow \mathbb{R}_{\geqslant 0}$ given by $h(K) = f(\min\{j \mid K \in \pi_j\})$ is a grading function for the hierarchy $\mathcal{H}_C$.

Proof. Since $\{x\} \in \pi_1 = \alpha_S$, it follows that $h(\{x\}) = 0$, so h satisfies the first condition of Definition 11.

Suppose that $H, K \in \mathcal{H}_C$ and $H \subset K$. If $\ell = \min\{j \mid H \in \pi_j\}$ it is impossible for K to be a block of a partition that precedes π_ℓ. Therefore, $\ell < \min\{j \mid K \in \pi_j\}$, so $h(H) < h(K)$, and $(\mathcal{H}_C, h)$ is indeed a graded hierarchy.

A graded hierarchy defines an ultrametric, as shown next.

Theorem 22. *Let $(\mathcal{H}, h)$ be a graded hierarchy on a finite set S. Define the function $d : S^2 \longrightarrow \mathbb{R}$ as $d(x, y) = \min\{h(U) \mid U \in \mathcal{H}$ and $\{x, y\} \subseteq U\}$ for $x, y \in S$. The mapping d is an ultrametric on S.*

Proof. Observe that for every $x, y \in S$ there exists a set $H \in \mathcal{H}$ such that $\{x, y\} \subseteq H$ because $S \in \mathcal{H}$.

It is immediate that $d(x, x) = 0$. Conversely, suppose that $d(x, y) = 0$. Then, there exists $H \in \mathcal{H}$ such that $\{x, y\} \subseteq H$ and $h(H) = 0$. If $x \neq y$, then $\{x\} \subset H$, hence $0 = h(\{x\}) < h(H)$, which contradicts the fact that $h(H) = 0$. Thus, $x = y$.

The symmetry of d is immediate.

To prove the ultrametric inequality, let $x, y, z \in S$, and suppose that $d(x, y) = p$, $d(x, z) = q$, and $d(z, y) = r$. There exist $H, K, L \in \mathcal{H}$ such that $\{x, y\} \subseteq H$, $h(H) = p$, $\{x, z\} \subseteq K$, $h(K) = q$, and $\{z, y\} \subseteq L$, $h(L) = r$. Since $K \cap L \neq \emptyset$ (because both sets contain z), we have either $K \subseteq L$ or $L \subseteq K$, so $K \cup L$ equals either K or L and, in either case, $K \cup L \in \mathcal{H}$. Since $\{x, y\} \subseteq K \cup L$, it follows that

$$d(x, y) \leqslant h(K \cup L) = \max\{h(K), H(L)\} = \max\{d(x, z), d(z, y)\},$$

which is the ultrametric inequality.

We refer to the ultrametric d whose existence is shown in Theorem 29 as the *ultrametric generated by the graded hierarchy* $(\mathcal{H}, h)$.

Example 32. The values of the ultrametric generated by the graded hierarchy $(\mathcal{H}, h)$ on the set S introduced in Example 36 are given in the following table:

d	s	t	u	v	w	x	y
s	0	3	3	5	7	7	7
t	3	0	3	5	7	7	7
u	3	3	0	5	7	7	7
v	5	5	5	0	7	7	7
w	7	7	7	7	0	4	6
x	7	7	7	7	4	0	6
y	7	7	7	7	6	6	0

The hierarchy introduced in Theorem 26 that is associated with an ultrametric space can be naturally equipped with a grading function, as shown next.

Theorem 23. *Let (S, d) be a finite ultrametric space. There exists a graded hierarchy $(\mathcal{H}, h)$ on S such that d is the ultrametric associated to $(\mathcal{H}, h)$.*

Proof. Let $\mathcal{H}$ be the collection of equivalence classes of the equivalences $\eta_r = \{(x, y) \in S^2 \mid d(x, y) \leqslant r\}$ defined by the ultrametric d on the finite set S, where the index r takes its values in the range R_d of the ultrametric d. Define $h(E) = \min\{r \in R_d \mid E \in S/\eta_r\}$ for every equivalence class E.

It is clear that $h(\{x\}) = 0$ because $\{x\}$ is an η_0-equivalence class for every $x \in S$.

Let $[x]_t$ be the equivalence class of x relative to the equivalence η_t.

Suppose that E and E' belong to the hierarchy and $E \subset E'$. We have $E = [x]_r$ and $E' = [x]_s$ for some $x \in X$. Since E is strictly included in E', there exists $z \in E' - E$ such that $d(x, z) \leqslant s$ and $d(x, z) > r$. This implies $r < s$. Therefore,

$$h(E) = \min\{r \in R_d \mid E \in S/\eta_r\} \leqslant \min\{s \in R_d \mid E' \in S/\eta_s\} = h(E'),$$

which proves that $(\mathcal{H}, h)$ is a graded hierarchy.

The ultrametric e generated by the graded hierarchy $(\mathcal{H}, h)$ is given by

$$\begin{aligned} e(x, y) &= \min\{h(B) \mid B \in \mathcal{H} \text{ and } \{x, y\} \subseteq B\} \\ &= \min\{r \mid (x, y) \in \eta_r\} = \min\{r \mid d(x, y) \leqslant r\} = d(x, y), \end{aligned}$$

for $x, y \in S$; in other words, we have $e = d$.

Example 33. Starting from the ultrametric on the set $S = \{s, t, u, v, w, x, y\}$ defined by the table given in Example 37, we obtain the following quotient sets:

Values of r	S/η_r
$[0, 3)$	$\{s\}, \{t\}, \{u\}, \{v\}, \{w\}, \{x\}, \{y\}$
$[3, 4)$	$\{s, t, u\}, \{v\}, \{w\}, \{x\}, \{y\}$
$[4, 5)$	$\{s, t, u\}, \{v\}, \{w, x\}, \{y\}$
$[5, 6)$	$\{s, t, u, v\}, \{w, x\}, \{y\}$
$[6, 7)$	$\{s, t, u, v\}, \{w, x, y\}$
$[7, \infty)$	$\{s, t, u, v, w, x, y\}$

We shall draw the tree of a graded hierarchy $(\mathcal{H}, h)$ using a special representation known as a *dendrogram*. In a dendrogram, an interior vertex K of the tree is represented by a horizontal line drawn at the height $h(K)$. For example, the dendrogram of the graded hierarchy of Example 36 is shown in Figure 7.5.

By Theorem 29, the value $d(x, y)$ of the ultrametric d generated by a hierarchy $\mathcal{H}$ is the smallest height of a set of a hierarchy that contains both x and y. This allows us to "read" the value of the ultrametric generated by $\mathcal{H}$ directly from the dendrogram of the hierarchy.

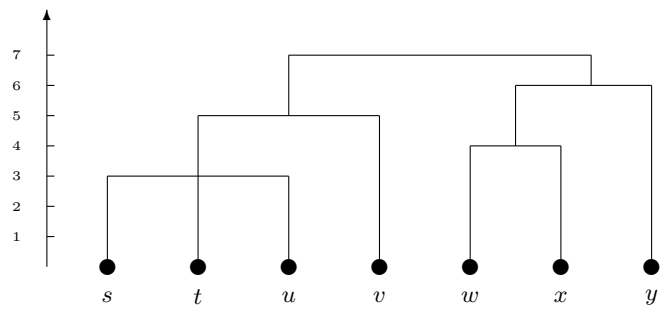

Fig. 7.2. Dendrogram of graded hierarchy of Example 36.

Example 34. For the graded hierarchy of Example 36, the ultrametric extracted from Figure 7.5 is clearly the same as the one that was obtained in Example 37.

The Poset of Ultrametrics

Let S be a set. Recall that we denoted the set of dissimilarities by $\mathcal{D}_S$. Define a partial order $\leqslant$ on $\mathcal{D}_S$ by $d \leqslant d'$ if $d(x, y) \leqslant d'(x, y)$ for every $x, y \in S$. It is easy to verify that $(\mathcal{D}_S, \leqslant)$ is a poset.

The set $\mathcal{U}_S$ of ultrametrics on S is a subset of $\mathcal{D}_S$.

Theorem 24. *Let d be a dissimilarity on a set S and let U_d be the set of ultrametrics $U_d = \{e \in \mathcal{U}_S \mid e \leqslant d\}$. The set U_d has a largest element in the poset $(\mathcal{D}_S, \leqslant)$.*

Proof. The set U_d is nonempty because the zero dissimilarity d_0 given by $d_0(x, y) = 0$ for every $x, y \in S$ is an ultrametric and $d_0 \leqslant d$.

Since the set $\{e(x, y) \mid e \in U_d\}$ has $d(x, y)$ as an upper bound, it is possible to define the mapping $e_1 : S^2 \longrightarrow \mathbb{R}_{\geqslant 0}$ as $e_1(x, y) = \sup\{e(x, y) \mid e \in U_d\}$ for $x, y \in S$. It is clear that $e \leqslant e_1$ for every ultrametric e. We claim that e_1 is an ultrametric on S.

We prove only that e_1 satisfies the ultrametric inequality. Suppose that there exist $x, y, z \in S$ such that e_1 violates the ultrametric inequality; that is,

$$\max\{e_1(x,z), e_1(z,y)\} < e_1(x,y).$$

This is equivalent to

$$\sup\{e(x,y) \mid e \in U_d\} > \max\{\sup\{e(x,z) \mid e \in U_d\}, \sup\{e(z,y) \mid e \in U_d\}\}.$$

Thus, there exists $\hat{e} \in U_d$ such that

$$\hat{e}(x,y) > \sup\{e(x,z) \mid e \in U_d\}, \text{ and } \hat{e}(x,y) > \sup\{e(z,y) \mid e \in U_d\}.$$

In particular, $\hat{e}(x,y) > \hat{e}(x,z)$ and $\hat{e}(x,y) > \hat{e}(z,y)$, which contradicts the fact that $\hat{e}$ is an ultrametric.

The ultrametric defined by Theorem 24 is known as the *maximal subdominant ultrametric for the dissimilarity d*.

The situation is not symmetric with respect to the infimum of a set of ultrametrics because, in general, the infimum of a set of ultrametrics is not necessarily an ultrametric.

For example, consider a three-element set $S = \{x, y, z\}$, four distinct nonnegative numbers a, b, c, d such that $a > b > c > d$ and the ultrametrics d and d' defined by the triangles shown in Figures 7.3(a) and (b), respectively. The dissimilarity d_0 defined by $d_0(u,v) = \min\{d(u,v), d'(u,v)\}$ for $u, v \in S$ is

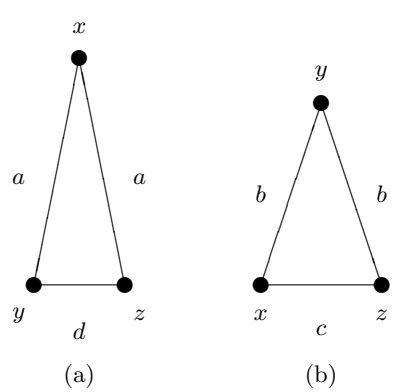

Fig. 7.3. Two ultrametrics on the set $\{x, y, z\}$.

given by

$$d_0(x,y) = b, d_0(y,z) = d, \text{ and } d_0(x,z) = c,$$

and d_0 is clearly not an ultrametric because the triangle xyz is not isosceles.

In what follows, we give an algorithm for computing the maximal subdominant ultrametric for a dissimilarity defined on a finite set S.

We define inductively an increasing sequence of partitions $\pi_1 \prec \pi_2 \prec \cdots$ and a sequence of dissimilarities $d_1, d_2, \ldots$ on the sets of blocks of $\pi_1, \pi_2, \ldots$, respectively.

For the initial phase, $\pi_1 = \alpha_S$ and $d_1(\{x\}, \{y\}) = d(x, y)$ for $x, y \in S$.

Suppose that d_i is defined on π_i. If $B, C \in \pi_i$ is a pair of blocks such that $d_i(B, C)$ has the smallest value, define the partition π_{i+1} by

$$\pi_{i+1} = (\pi_i - \{B, C\}) \cup \{B \cup C\}.$$

In other words, to obtain π_{i+1}, we replace two of the closest blocks B and C, of π_i (in terms of d_i) with new block $B \cup C$. Clearly, $\pi_i \prec \pi_{i+1}$ in $PART(S)$ for $i \geqslant 1$.

The collection of blocks of the partitions π_i forms a hierarchy $\mathcal{H}_d$ on the set S. The dissimilarity d_{i+1} is given by

$$d_{i+1}(U, V) = \min\{d(x, y) \mid x \in U, y \in V\} \tag{7.2}$$

for $U, V \in \pi_{i+1}$.

We introduce a grading function h_d on the hierarchy defined by this chain of partitions starting from the dissimilarity d. The definition is done for the blocks of the partitions π_i by induction on i.

For $i = 1$ the blocks of the partition π_1 are singletons; in this case we define $h_d(\{x\}) = 0$ for $x \in S$.

Suppose that h_d is defined on the blocks of π_i, and let D be the block of π_{i+1} that is generated by fusing the blocks B and C of π_i. All other blocks of π_{i+1} coincide with the blocks of π_i. The value of the function h_d for the new block D is given by $h_d(D) = \min\{d(x, y) \mid x \in B, y \in C\}$. It is clear that h_d satisfies the first condition of Definition 11.

For a set U of $\mathcal{H}_d$, define $p_U = \min\{i \mid U \in \pi_i\}$ and $q_U = \max\{i \mid U \in \pi_i\}$. Note that p_U is the first index i such that U is a block of π_i, and q_U is the last i such that U is a block of π_i. If $H, K \in \mathcal{H}_\Gamma$ and $H \subseteq K$, this means that both H and K are blocks of some partitions π_h and π_k and we have

$$p_H \leqslant q_H \leqslant p_K \leqslant q_K,$$

so $q_H \leqslant p_K$.

The construction of the sequence of partitions implies that there are $H_0, H_1 \in \pi_{p_H - 1}$ and $K_0, K_1 \in \pi_{p_K - 1}$ such that $H = H_0 \cup H_1$ and $K = K_0 \cup K_1$. Therefore,

$$h_d(H) = \min\{d(x, y) \mid x \in H_0, y \in H_1\},$$
$$h_d(K) = \min\{d(x, y) \mid x \in K_0, y \in K_1\}.$$

Since H_0 and H_1 were fused (to produce the block H of the partition π_{p_H}) before K_0 and K_1 were (to produce the block K of the partition π_{p_K}), it follows that $h_d(H) < h_d(K)$.

By Theorem 29, the graded hierarchy $(\mathcal{H}_d, h_d)$ defines an ultrametric; we denote this ultrametric by e and will prove that e is the maximal subdominant ultrametric for d. Recall that e is given by

$$e(x,y) = \min\{h_d(W) \mid \{x,y\} \subseteq W\}$$

and that $h_d(W)$ is the least value of $d(u,v)$ such that $u \in U, v \in V$ if $W \in \pi_{p_W}$ is obtained by fusing the blocks U and V of π_{p_W-1}. The definition of $e(x,y)$ implies that we have neither $\{x,y\} \subseteq U$ nor $\{x,y\} \subseteq V$. Thus, we have either $x \in U$ and $y \in V$ or $x \in V$ and $y \in U$. Thus, $e(x,y) \leqslant d(x,y)$.

We now prove that:

$$e(x,y) = \min\{amp_d(\boldsymbol{s}) \mid \boldsymbol{s} \in S(x,y)\}$$

for $x, y \in S$.

Let D be the minimal set in $\mathcal{H}_d$ that includes $\{x,y\}$. Then, $D = B \cup C$, where B and C are two disjoint sets of $\mathcal{H}_d$ such that $x \in B$ and $y \in C$. If $\boldsymbol{s}$ is a sequence included in D, then there are two consecutive components of $\boldsymbol{s}$, s_k and s_{k+1}, such that $s_k \in B$ and $s_{k+1} \in C$. This implies

$$\begin{aligned} e(x,y) &= \min\{d(u,v) \mid u \in B, v \in C\} \\ &\leqslant d(s_k, s_{k+1}) \\ &\leqslant amp_d(\boldsymbol{s}). \end{aligned}$$

If $\boldsymbol{s}$ is not included in D, let s_q and s_{q+1} be two consecutive components of $\boldsymbol{s}$ such that $s_q \in D$ and $s_{q+1} \notin D$. Let E be the smallest set of $\mathcal{H}_d$ that includes $\{s_q, s_{q+1}\}$. We have $D \subseteq E$ (because $s_k \in D \cap E$) and therefore $h_d(D) \leqslant h_d(E)$. If E is obtained as the union of two disjoint sets E' and E'' of $\mathcal{H}_d$ such that $s_k \in E'$ and $s_{k+1} \in E''$, we have $D \subseteq E'$. Consequently,

$$h_d(E) = \min\{d(u,v) \mid u \in E', v \in E''\} \leqslant d(s_k, s_{k+1}),$$

which implies

$$e(x,y) = h_d(D) \leqslant h_d(E) \leqslant d(s_k, s_{k+1}) \leqslant amp_d(\boldsymbol{s}).$$

Therefore, we conclude that $e(x,y) \leqslant amp_d(\boldsymbol{s})$ for every $\boldsymbol{s} \in S(x,y)$.

We now show that there is a sequence $\boldsymbol{w} \in S(x,y)$ such that $e(x,y) \geqslant amp_d(\boldsymbol{w})$, which implies the equality $e(x,y) = amp_d(\boldsymbol{w})$. To this end, we prove that for every $D \in \pi_k \subseteq \mathcal{H}_d$ there exists $\boldsymbol{w} \in S(x,y)$ such that $amp_d(\boldsymbol{w}) \leqslant h_d(D)$. The argument is by induction on k.

For $k = 1$, the statement obviously holds. Suppose that it holds for $1, \ldots, k-1$, and let $D \in \pi_k$. The set D belongs to π_{k-1} or D is obtained by fusing the blocks B, C of π_{k-1}. In the first case, the statement holds by inductive hypothesis. The second case has several subcases:

(i) If $\{x,y\} \subseteq B$, then by the inductive hypothesis, there exists a sequence $\boldsymbol{u} \in S(x,y)$ such that $amp_d(\boldsymbol{u}) \leqslant h_d(B) \leqslant h_d(D) = e(x,y)$.

(ii) The case $\{x, y\} \subseteq C$ is similar to the first case.
(iii) If $x \in B$ and $y \in C$, there exist $u, v \in D$ such that $d(u, v) = h_d(D)$. By the inductive hypothesis, there is a sequence $\boldsymbol{u} \in S(x, u)$ such that $amp_d(\boldsymbol{u}) \leqslant h_d(B)$ and there is a sequence $\boldsymbol{v} \in S(v, y)$ such that $amp_d(\boldsymbol{v}) \leqslant h_d(C)$. This allows us to consider the sequence $\boldsymbol{w}$ obtained by concatenating the sequences $\boldsymbol{u}, (u, v), \boldsymbol{v}$; clearly, $\boldsymbol{w} \in S(x, y)$ and $amp_d(\boldsymbol{w}) = \max\{amp_d(\boldsymbol{u}), d(u, v), amp_d(\boldsymbol{v})\} \leqslant h_d(D)$.

To complete the argument, we need to show that if e' is another ultrametric such that $e(x, y) \leqslant e'(x, y) \leqslant d(x, y)$, then $e(x, y) = e'(x, y)$ for every $x, y \in S$. By the previous argument, there exists a sequence $\boldsymbol{s} = (s_0, \ldots, s_n) \in S(x, y)$ such that $amp_d(\boldsymbol{s}) = e(x, y)$. Since $e'(x, y) \leqslant d(x, y)$ for every $x, y \in S$, it follows that $e'(x, y) \leqslant amp_d(\boldsymbol{s}) = e(x, y)$. Thus, $e(x, y) = e'(x, y)$ for every $x, y \in S$, which means that $e = e'$. This concludes our argument.

7.3 Hierarchies on Sets

Definition 10. *Let S be a set. A* hierarchy *on S is a collection of sets $\mathcal{H} \subseteq \mathcal{P}(S)$ that satisfies the following conditions:*
(i) *the members of $\mathcal{H}$ are nonempty sets;*
(ii) *$S \in \mathcal{H}$;*
(iii) *for every $x \in S$, we have $\{x\} \in \mathcal{H}$;*
(iv) *if $H, H' \in \mathcal{H}$ and $H \cap H' \neq \emptyset$, then we have either $H \subseteq H'$ or $H' \subseteq H$.*

A standard technique for constructing a hierarchy on a set S starts with a rooted tree $(\mathcal{T}, v_0)$ whose nodes are labeled by subsets of the set S. Let V be the set of vertices of the tree $\mathcal{T}$. The function $\mu : V \longrightarrow \mathcal{P}(S)$, which gives the label $\mu(v)$ of each node $v \in V$, is defined as follows:
(i) the tree $\mathcal{T}$ has $|S|$ leaves, and each leaf v is labeled by a distinct singleton $\mu(v) = \{x\}$ for $x \in S$;
(ii) if an interior vertex v of the tree has the descendants $v_1, v_2, \ldots, v_n$, then $\mu(v) = \bigcup_{i=1}^{n} \mu(v_i)$.

The set of labels $\mathcal{H}_{\mathcal{T}}$ of the rooted tree $(\mathcal{T}, v_0)$ forms a hierarchy on S. Indeed, note that each singleton $\{x\}$ is a label of a leaf. An easy argument by induction on the height of the tree shows that every vertex is labeled by the set of labels of the leaves that descend from that vertex. Therefore, the root v_0 of the tree is labeled by S.

Suppose that H, H' are labels of the nodes u, v of $\mathcal{T}$, respectively. If $H \cap H' \neq \emptyset$, then the vertices u, v have a common descendant. In a tree, this can take place only if u is a descendant of v or v is a descendant of u; that is, only if $H \subseteq H'$, or $H' \subseteq H$, respectively. This gives the desired conclusion.

Example 35. Let $S = \{s, t, u, v, w, x, y\}$ and let $\mathcal{T}$ be a tree whose vertices are labeled as shown in Figure 7.4. It is easy to verify that the family of subsets of S that label the nodes of $\mathcal{T}$,

$$\mathcal{H} = \{\{s\}, \{t\}, \{u\}, \{v\}, \{w\}, \{x\}, \{y\}, \\ \{s,t,u\}, \{w,x\}, \{s,t,u,v\}, \{w,x,y\}, \{s,t,u,v,w,x,y\}\}$$

is a hierarchy on the set S.

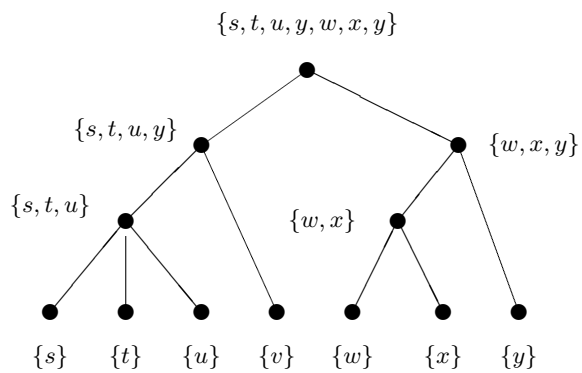

Fig. 7.4. Tree labeled by subsets of S.

Chains of partitions defined on a set generate hierarchies, as we show next.

Theorem 25. *Let S be a set and let $C = (\pi_1, \pi_2, \ldots, \pi_n)$ be an increasing chain of partitions $(PART(S), \leqslant)$ such that $\pi_1 = \alpha_S$ and $\pi_n = \omega_S$. Then, the collection $\mathcal{H}_C = \bigcup_{i=1}^{n} \pi_i$ that consists of the blocks of all partitions in the chain is a hierarchy on S.*

Proof. The blocks of any of the partitions are nonempty sets, so $\mathcal{H}_C$ satisfies the first condition of Definition 10.

We have $S \in \mathcal{H}_C$ because S is the unique block of $\pi_n = \omega_S$. Also, since all singletons $\{x\}$ are blocks of $\alpha_S = \pi_1$, it follows that $\mathcal{H}_C$ satisfies the second and the third conditions of Definition 10. Finally, let H and H' be two sets of $\mathcal{H}_C$ such that $H \cap H' \neq \emptyset$. Because of this condition, it is clear that these two sets cannot be blocks of the same partition. Thus, there exist two partitions π_i and π_j in the chain such that $H \in \pi_i$ and $H' \in \pi_j$. Suppose that $i < j$. Since every block of π_j is a union of blocks of π_i, H' is a union of blocks of π_i and $H \cap H' \neq \emptyset$ means that H is one of these blocks. Thus, $H \subseteq H'$. If $j > i$, we obtain the reverse inclusion. This allows us to conclude that $\mathcal{H}_C$ is indeed a hierarchy.

Theorem 25 can be stated in terms of chains of equivalences; we give the following alternative formulation for convenience.

Theorem 26. *Let S be a finite set and let $(\rho_1, \ldots, \rho_n)$ be a chain of equivalence relations on S such that $\rho_1 = \iota_S$ and $\rho_n = \theta_S$. Then, the collection*

of blocks of the equivalence relations ρ_r (that is, the set $\bigcup_{1\leqslant r\leqslant n} S/\rho_r$) is a hierarchy on S.

Proof. The proof is a mere restatement of the proof of Theorem 25.

Define the relation "$\prec$" on a hierarchy $\mathcal{H}$ on S by $H \prec K$ if $H, K \in \mathcal{H}$, $H \subset K$, and there is no set $L \in \mathcal{H}$ such that $H \subset L \subset K$.

Lemma 2. *Let $\mathcal{H}$ be a hierarchy on a finite set S and let $L \in \mathcal{H}$. The collection $\mathcal{P}_L = \{H \in \mathcal{H} \mid H \prec L\}$ is a partition of the set L.*

Proof. We claim that $L = \bigcup \mathcal{P}_L$. Indeed, it is clear that $\bigcup \mathcal{P}_L \subseteq L$.

Conversely, suppose that $z \in L$ but $z \notin \bigcup \mathcal{P}_L$. Since $\{z\} \in \mathcal{H}$ and there is no $K \in \mathcal{P}_L$ such that $z \in K$, it follows that $\{z\} \in \mathcal{P}_L$, which contradicts the assumption that $z \notin \bigcup \mathcal{P}_L$. This means that $L = \bigcup \mathcal{P}_L$.

Let $K_0, K_1 \in \mathcal{P}_L$ be two distinct sets. These sets are disjoint since otherwise we would have either $K_0 \subset K_1$ or $K_1 \subset K_0$, and this would contradict the definition of $\mathcal{P}_L$.

Theorem 27. *Let $\mathcal{H}$ be a hierarchy on a set S. The graph of the relation $\prec$ on $\mathcal{H}$ is a tree whose root is S; its leaves are the singletons $\{x\}$ for every $x \in S$.*

Proof. Since $\prec$ is an antisymmetric relation on $\mathcal{H}$, it is clear that the graph $(\mathcal{H}, \prec)$ is acyclic. Moreover, for each set $K \in \mathcal{H}$, there is a unique path that joins K to S, so the graph is indeed a rooted tree.

Definition 11. *Let $\mathcal{H}$ be a hierarchy on a set S. A* grading function *for $\mathcal{H}$ is a function $h : \mathcal{H} \longrightarrow \mathbb{R}$ that satisfies the following conditions:*
(i) *$h(\{x\}) = 0$ for every $x \in S$, and*
(ii) *if $H, K \in \mathcal{H}$ and $H \subset K$, then $h(H) < h(K)$.*

If h is a grading function for a hierarchy $\mathcal{H}$, the pair $(\mathcal{H}, h)$ is a graded hierarchy*.*

Example 36. For the hierarchy $\mathcal{H}$ defined in Example 35 on the set $S = \{s, t, u, v, w, x, y\}$, the function $h : \mathcal{H} \longrightarrow \mathbb{R}$ given by

$$\begin{aligned}&h(\{s\}) = h(\{t\}) = h(\{u\}) = h(\{v\}) = h(\{w\}) = h(\{x\}) = h(\{y\}) = 0,\\ &h(\{s,t,u\}) = 3, h(\{w,x\}) = 4, h(\{s,t,u,v\}) = 5, h(\{w,x,y\}) = 6,\\ &h(\{s,t,u,v,w,x,y\}) = 7,\end{aligned}$$

is a grading function and the pair $(\mathcal{H}, h)$ is a graded hierarchy on S.

Theorem 25 can be extended to graded hierarchies.

Theorem 28. *Let S be a finite set and let $C = (\pi_1, \pi_2, \ldots, \pi_n)$ be an increasing chain of partitions $(PART(S), \leqslant)$ such that $\pi_1 = \alpha_S$ and $\pi_n = \omega_S$.*

If $f : \{1, \ldots, n\} \longrightarrow \mathbb{R}_{\geqslant 0}$ is a function such that $f(1) = 0$, then the function $h : \mathcal{H}_C \longrightarrow \mathbb{R}_{\geqslant 0}$ given by $h(K) = f(\min\{j \mid K \in \pi_j\})$ is a grading function for the hierarchy $\mathcal{H}_C$.

Proof. Since $\{x\} \in \pi_1 = \alpha_S$, it follows that $h(\{x\}) = 0$, so h satisfies the first condition of Definition 11.

Suppose that $H, K \in \mathcal{H}_C$ and $H \subset K$. If $\ell = \min\{j \mid H \in \pi_j\}$ it is impossible for K to be a block of a partition that precedes π_ℓ. Therefore, $\ell < \min\{j \mid K \in \pi_j\}$, so $h(H) < h(K)$, and $(\mathcal{H}_C, h)$ is indeed a graded hierarchy.

A graded hierarchy defines an ultrametric, as shown next.

Theorem 29. *Let $(\mathcal{H}, h)$ be a graded hierarchy on a finite set S. Define the function $d : S^2 \longrightarrow \mathbb{R}$ as $d(x, y) = \min\{h(U) \mid U \in \mathcal{H}$ and $\{x, y\} \subseteq U\}$ for $x, y \in S$. The mapping d is an ultrametric on S.*

Proof. Observe that for every $x, y \in S$ there exists a set $H \in \mathcal{H}$ such that $\{x, y\} \subseteq H$ because $S \in \mathcal{H}$.

It is immediate that $d(x, x) = 0$. Conversely, suppose that $d(x, y) = 0$. Then, there exists $H \in \mathcal{H}$ such that $\{x, y\} \subseteq H$ and $h(H) = 0$. If $x \neq y$, then $\{x\} \subset H$, hence $0 = h(\{x\}) < h(H)$, which contradicts the fact that $h(H) = 0$. Thus, $x = y$.

The symmetry of d is immediate.

To prove the ultrametric inequality, let $x, y, z \in S$, and suppose that $d(x, y) = p$, $d(x, z) = q$, and $d(z, y) = r$. There exist $H, K, L \in \mathcal{H}$ such that $\{x, y\} \subseteq H$, $h(H) = p$, $\{x, z\} \subseteq K$, $h(K) = q$, and $\{z, y\} \subseteq L$, $h(L) = r$. Since $K \cap L \neq \emptyset$ (because both sets contain z), we have either $K \subseteq L$ or $L \subseteq K$, so $K \cup L$ equals either K or L and, in either case, $K \cup L \in \mathcal{H}$. Since $\{x, y\} \subseteq K \cup L$, it follows that

$$d(x, y) \leqslant h(K \cup L) = \max\{h(K), H(L)\} = \max\{d(x, z), d(z, y)\},$$

which is the ultrametric inequality.

We refer to the ultrametric d whose existence is shown in Theorem 29 as the *ultrametric generated by the graded hierarchy* $(\mathcal{H}, h)$.

Example 37. The values of the ultrametric generated by the graded hierarchy $(\mathcal{H}, h)$ on the set S introduced in Example 36 are given in the following table:

d	s	t	u	v	w	x	y
s	0	3	3	5	7	7	7
t	3	0	3	5	7	7	7
u	3	3	0	5	7	7	7
v	5	5	5	0	7	7	7
w	7	7	7	7	0	4	6
x	7	7	7	7	4	0	6
y	7	7	7	7	6	6	0

The hierarchy introduced in Theorem 26 that is associated with an ultrametric space can be naturally equipped with a grading function, as shown next.

Theorem 30. *Let (S, d) be a finite ultrametric space. There exists a graded hierarchy $(\mathcal{H}, h)$ on S such that d is the ultrametric associated to $(\mathcal{H}, h)$.*

Proof. Let $\mathcal{H}$ be the collection of equivalence classes of the equivalences $\eta_r = \{(x, y) \in S^2 \mid d(x, y) \leqslant r\}$ defined by the ultrametric d on the finite set S, where the index r takes its values in the range R_d of the ultrametric d. Define $h(E) = \min\{r \in R_d \mid E \in S/\eta_r\}$ for every equivalence class E.

It is clear that $h(\{x\}) = 0$ because $\{x\}$ is an η_0-equivalence class for every $x \in S$.

Let $[x]_t$ be the equivalence class of x relative to the equivalence η_t.

Suppose that E and E' belong to the hierarchy and $E \subset E'$. We have $E = [x]_r$ and $E' = [x]_s$ for some $x \in X$. Since E is strictly included in E', there exists $z \in E' - E$ such that $d(x, z) \leqslant s$ and $d(x, z) > r$. This implies $r < s$. Therefore,

$$h(E) = \min\{r \in R_d \mid E \in S/\eta_r\} \leqslant \min\{s \in R_d \mid E' \in S/\eta_s\} = h(E'),$$

which proves that $(\mathcal{H}, h)$ is a graded hierarchy.

The ultrametric e generated by the graded hierarchy $(\mathcal{H}, h)$ is given by

$$\begin{aligned} e(x, y) &= \min\{h(B) \mid B \in \mathcal{H} \text{ and } \{x, y\} \subseteq B\} \\ &= \min\{r \mid (x, y) \in \eta_r\} = \min\{r \mid d(x, y) \leqslant r\} = d(x, y), \end{aligned}$$

for $x, y \in S$; in other words, we have $e = d$.

Example 38. Starting from the ultrametric on the set $S = \{s, t, u, v, w, x, y\}$ defined by the table given in Example 37, we obtain the following quotient sets:

Values of r	S/η_r
$[0, 3)$	$\{s\}, \{t\}, \{u\}, \{v\}, \{w\}, \{x\}, \{y\}$
$[3, 4)$	$\{s, t, u\}, \{v\}, \{w\}, \{x\}, \{y\}$
$[4, 5)$	$\{s, t, u\}, \{v\}, \{w, x\}, \{y\}$
$[5, 6)$	$\{s, t, u, v\}, \{w, x\}, \{y\}$
$[6, 7)$	$\{s, t, u, v\}, \{w, x, y\}$
$[7, \infty)$	$\{s, t, u, v, w, x, y\}$

We shall draw the tree of a graded hierarchy $(\mathcal{H}, h)$ using a special representation known as a *dendrogram*. In a dendrogram, an interior vertex K of the tree is represented by a horizontal line drawn at the height $h(K)$. For example, the dendrogram of the graded hierarchy of Example 36 is shown in Figure 7.5.

By Theorem 29, the value $d(x, y)$ of the ultrametric d generated by a hierarchy $\mathcal{H}$ is the smallest height of a set of a hierarchy that contains both x and y. This allows us to "read" the value of the ultrametric generated by $\mathcal{H}$ directly from the dendrogram of the hierarchy.

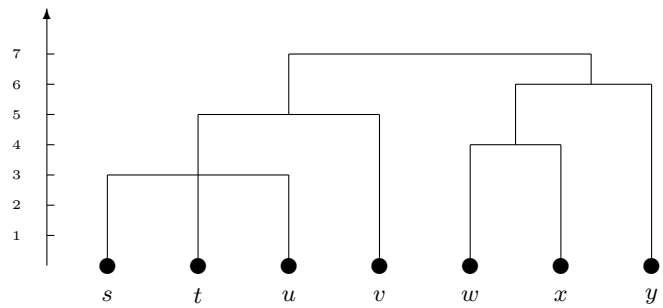

Fig. 7.5. Dendrogram of graded hierarchy of Example 36.

Example 39. For the graded hierarchy of Example 36, the ultrametric extracted from Figure 7.5 is clearly the same as the one that was obtained in Example 37.

7.4 Hierarchical Clustering

Hierarchical clustering is a recursive process that begins with a metric space of objects (S, d) and results in a chain of partitions of the set of objects. In each of the partitions, similar objects belong to the same block and objects that belong to distinct blocks tend to be dissimilar.

In agglomerative hierarchical clustering, the construction of this chain begins with the unit partition $\pi^1 = \alpha_S$. If the partition constructed at step k is

$$\pi^k = \{U_1^k, \ldots, U_{m_k}^k\},$$

then two distinct blocks U_p^k and U_q^k of this partition are selected using a *selection criterion*. These blocks are fused and a new partition

$$\pi^{k+1} = \{U_1^k, \ldots, U_{p-1}^k, U_{p+1}^k, \ldots, U_{q-1}^k, U_{q+1}^k, \ldots, U_p^k \cup U_q^k\}$$

is formed. Clearly, we have $\pi^k \prec \pi^{k+1}$. The process must end because the poset $(PART(S), \leqslant)$ is of finite height. The algorithm halts when the one-block partition ω_S is reached.

If two blocks U and V of a partition π are fused into a new block W to yield a new partition π' that covers π, then the variation of the sum of squared errors is given by

$$\begin{aligned} sse(\pi') - sse(\pi) = &\sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) | \boldsymbol{o} \in U \cup V\} \\ &- \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_U) | \boldsymbol{o} \in U\} - \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_V) | \boldsymbol{o} \in V\}. \end{aligned}$$

The centroid of the new cluster W is given by

$$\boldsymbol{c}_W = \frac{1}{|W|} \sum\{\boldsymbol{o} | \boldsymbol{o} \in W\} = \frac{|U|}{|W|}\boldsymbol{c}_U + \frac{|V|}{|W|}\boldsymbol{c}_V.$$

This allows us to evaluate the increase in the sum of squared errors:

$$\begin{aligned} sse(\pi') - sse(\pi) &= \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) \mid \boldsymbol{o} \in U \cup V\} \\ &\quad - \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_U) \mid \boldsymbol{o} \in U\} - \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_V) \mid \boldsymbol{o} \in V\} \\ &= \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) - d^2(\boldsymbol{o}, \boldsymbol{c}_U) \mid \boldsymbol{o} \in U\} \\ &\quad + \sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) - d^2(\boldsymbol{o}, \boldsymbol{c}_V) \mid \boldsymbol{o} \in V\}. \end{aligned}$$

Observe that:

$$\begin{aligned} &\sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) - d^2(\boldsymbol{o}, \boldsymbol{c}_U) \mid \boldsymbol{o} \in U\} \\ &= \sum_{\boldsymbol{o} \in U} ((\boldsymbol{o} - \boldsymbol{c}_W)'(\boldsymbol{o} - \boldsymbol{c}_W) - (\boldsymbol{o} - \boldsymbol{c}_U)'(\boldsymbol{o} - \boldsymbol{c}_U)) \\ &= |U|(\boldsymbol{c}_W^2 - \boldsymbol{c}_U^2) + 2(\boldsymbol{c}_U - \boldsymbol{c}_W)' \sum_{\boldsymbol{o} \in U} \boldsymbol{o} \\ &= |U|(\boldsymbol{c}_W^2 - \boldsymbol{c}_U^2) + 2|U|(\boldsymbol{c}_U - \boldsymbol{c}_W)'\boldsymbol{c}_U \\ &= |U|(\boldsymbol{c}_W - \boldsymbol{c}_U)^2. \end{aligned}$$

Using the equality $\boldsymbol{c}_W - \boldsymbol{c}_U = \frac{|U|}{|W|}\boldsymbol{c}_U + \frac{|V|}{|W|}\boldsymbol{c}_V - \boldsymbol{c}_U = \frac{|V|}{|W|}(\boldsymbol{c}_V - \boldsymbol{c}_U)$, we obtain $\sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) - d^2(\boldsymbol{o}, \boldsymbol{c}_U) \mid \boldsymbol{o} \in U\} = \frac{|U||V|^2}{|W|^2}(\boldsymbol{c}_V - \boldsymbol{c}_U)^2$.

Similarly, we have

$$\sum\{d^2(\boldsymbol{o}, \boldsymbol{c}_W) - d^2(\boldsymbol{o}, \boldsymbol{c}_V) \mid \boldsymbol{o} \in V\} = \frac{|U|^2|V|}{|W|^2}(\boldsymbol{c}_V - \boldsymbol{c}_U)^2,$$

so,

$$sse(\pi') - sse(\pi) = \frac{|U||V|}{|W|}(\boldsymbol{c}_V - \boldsymbol{c}_U)^2. \tag{7.3}$$

In each phase of hierarchical clustering two of the "closest" clusters are merged. The notion of closest clusters is dependent on the specific dissimilarity between clusters considered in each variant of the clustering algorithm. If U and V are two clusters, the dissimilarity between them is be defined using one of the following real-valued, two-argument functions defined on the set of subsets of S:

$$\begin{aligned}
sl(U,V) &= \min\{d(u,v)|u \in U, v \in V\};\\
cl(U,V) &= \max\{d(u,v)|u \in U, v \in V\};\\
gav(U,V) &= \frac{\sum\{d(u,v)|u \in U, v \in V\}}{|U| \cdot |V|};\\
cen(U,V) &= (\boldsymbol{c}_U - \boldsymbol{c}_V)^2;\\
ward(U,V) &= \frac{|U||V|}{|U|+|V|}\left(\boldsymbol{c}_V - \boldsymbol{c}_U\right)^2.
\end{aligned}$$

The names of the functions *sl*, *cl*, *gav*, and *cen* defined above are acronyms of the terms "single link", "complete link", "group average", and "centroid", respectively. They are linked to variants of the hierarchical clustering algorithms that we discuss later. Note that in the case of the *ward* function the value equals the increase in the sum of the square errors when the clusters U, V are replaced with their union.

The specific selection criterion for fusing blocks defines the clustering algorithm. All algorithms store the dissimilarities between the current clusters $\pi^k = \{U_1^k, \ldots, U_{m_k}^k\}$ in an $m_k \times m_k$-matrix $D^k = (d_{ij}^k)$, where d_{ij}^k is the dissimilarity between the clusters U_i^k and U_j^k. As new clusters are created by merging two existing clusters, the distance matrix must be adjusted to reflect the dissimilarities between the new cluster and existing clusters.

The general form of the algorithm is shown as Algorithm 7.4.1.

Algorithm 7.4.1: Agglomerative Clustering Algorithm

Data: the initial dissimilarity matrix D^1
Result: the cluster hierarchy on the set of objects S, where $|S| = n$

```
k = 1;
initialize clustering: π^1 = α_S;
while π^k contains more than one block do
    merge a pair of two of the closest clusters;
    output new cluster;
    k++;
    compute the dissimilarity matrix D^k;
end
```

To evaluate the space and time complexity of hierarchical clustering, note that the algorithm must handle the matrix of the dissimilarities between objects, and this is a symmetric $n \times n$-matrix having all elements on its main diagonal equal to 0; in other words, the algorithm needs to store $\frac{n(n-1)}{2}$ numbers. To keep track of the clusters, an extra space that does not exceed $n-1$ is required. Thus, the total space required is $O(n^2)$.

The computation of the dissimilarity between a new cluster and existing clusters is described next.

Theorem 31. *Let U and V be two clusters of the clustering π that are joined into a new cluster W. Then, if $Q \in \pi - \{U, V\}$, we have*

$$\begin{aligned}
sl(W,Q) &= \frac{1}{2}sl(U,Q) + \frac{1}{2}sl(V,Q) - \frac{1}{2}\Big|sl(U,Q) - sl(V,Q)\Big|;\\
cl(W,Q) &= \frac{1}{2}cl(U,Q) + \frac{1}{2}cl(V,Q) + \frac{1}{2}\Big|cl(U,Q) - cl(V,Q)\Big|;\\
gav(W,Q) &= \frac{|U|}{|U|+|V|}gav(U,Q) + \frac{|V|}{|U|+|V|}gav(V,Q);\\
cen(W,Q) &= \frac{|U|}{|U|+|V|}cen(U,Q) + \frac{|V|}{|U|+|V|}cen(V,Q)\\
&\quad - \frac{|U||V|}{(|U|+|V|)^2}cen(U,V);\\
ward(W,Q) &= \frac{|U|+|Q|}{|U|+|V|+|Q|}ward(U,Q) + \frac{|V|+|Q|}{|U|+|V|+|Q|}ward(V,Q)\\
&\quad - \frac{|Q|}{|U|+|V|+|Q|}ward(U,V).
\end{aligned}$$

Proof. The first two equalities follow from the fact that

$$\begin{aligned}
\min\{a,b\} &= \frac{1}{2}(a+b) - \frac{1}{2}|a-b|,\\
\max\{a,b\} &= \frac{1}{2}(a+b) + \frac{1}{2}|a-b|,
\end{aligned}$$

for every $a, b \in \mathbb{R}$.

For the third equality, we have

$$\begin{aligned}
gav(W,Q) &= \frac{\sum\{d(w,q)|w \in W, q \in Q\}}{|W|\cdot|Q|}\\
&= \frac{\sum\{d(u,q)|u \in U, q \in Q\}}{|W|\cdot|Q|} + \frac{\sum\{d(v,q)|v \in V, q \in Q\}}{|W|\cdot|Q|}\\
&= \frac{|U|}{|W|}\frac{\sum\{d(u,q)|u \in U, q \in Q\}}{|U|\cdot|Q|} + \frac{|V|}{|W|}\frac{\sum\{d(v,q)|v \in V, q \in Q\}}{|V|\cdot|Q|}\\
&= \frac{|U|}{|U|+|V|}gav(U,Q) + \frac{|V|}{|U|+|V|}gav(V,Q).
\end{aligned}$$

The equality involving the function *cen* is immediate. The last equality can be easily translated into

$$\begin{aligned}
&\frac{|Q||W|}{|Q|+|W|}\left(\boldsymbol{c}_Q-\boldsymbol{c}_W\right)^2\\
&=\frac{|U|+|Q|}{|U|+|V|+|Q|}\frac{|U||Q|}{|U|+|Q|}\left(\boldsymbol{c}_Q-\boldsymbol{c}_U\right)^2\\
&\quad+\frac{|V|+|Q|}{|U|+|V|+|Q|}\frac{|V||Q|}{|V|+|Q|}\left(\boldsymbol{c}_Q-\boldsymbol{c}_V\right)^2\\
&\quad-\frac{|Q|}{|U|+|V|+|Q|}\frac{|U||V|}{|U|+|V|}\left(\boldsymbol{c}_V-\boldsymbol{c}_U\right)^2,
\end{aligned}$$

which can be verified replacing $|W|=|U|+|V|$ and $\boldsymbol{c}_W=\frac{|U|}{|W|}\boldsymbol{c}_U+\frac{|V|}{|W|}\boldsymbol{c}_V$. □

The equalities contained by Theorem 31 are often presented as a single equality involving several coefficients.

Corollary 3 (The Lance-Williams Formula). *Let U and V be two clusters of the clustering π that are joined into a new cluster W. Then, if $Q\in\pi-\{U,V\}$, the dissimilarity between W and Q can be expressed as*

$$d(W,Q)=a_Ud(U,Q)+a_Vd(V,Q)+bd(U,V)+c|d(U,Q)-d(V,Q)|,$$

where the coefficients a_U,a_V,b,c are given by the following table:

Function	a_U	a_V	b	c
sl	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
cl	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
gav	$\frac{\lvert U\rvert}{\lvert U\rvert+\lvert V\rvert}$	$\frac{\lvert V\rvert}{\lvert U\rvert+\lvert V\rvert}$	0	0
cen	$\frac{\lvert U\rvert}{\lvert U\rvert+\lvert V\rvert}$	$\frac{\lvert V\rvert}{\lvert U\rvert+\lvert V\rvert}$	$-\frac{\lvert U\rvert\lvert V\rvert}{(\lvert U\rvert+\lvert V\rvert)^2}$	0
ward	$\frac{\lvert U\rvert+\lvert Q\rvert}{\lvert U\rvert+\lvert V\rvert+\lvert Q\rvert}$	$\frac{\lvert V\rvert+\lvert Q\rvert}{\lvert U\rvert+\lvert V\rvert+\lvert Q\rvert}$	$-\frac{\lvert Q\rvert}{\lvert U\rvert+\lvert V\rvert+\lvert Q\rvert}$	0

Proof. This statement is an immediate consequence of Theorem 31. □

The variant of the algorithm that makes use of the function *sl* is known as the *single-link* clustering. It tends to favor elongated clusters.

The *group average method*, which makes use of the *gav* function generates an intermediate approach between the single-link and the complete-link method. What the methods mentioned so far have in common is the *monotonicity property* expressed by the following statement.

Theorem 32. *Let (S,d) be a finite metric space and let $D^1,\ldots,D^m$ be the sequence of matrices constructed by any of the first three hierarchical methods (single, complete, or average link), where $m=|S|$. If μ_i is the smallest entry of the matrix D^i for $1\leqslant i\leqslant m$, then $\mu_1\leqslant\mu_2\leqslant\cdots\leqslant\mu_m$. In other words, the dissimilarity between clusters that are merged at each step is nondecreasing.*

Proof. Let D^{j+1} be the matrix is obtained from the matrix D^j by merging the clusters C_p and C_q that correspond to the lines p and q and to columns p, q of D^j. This happens because $d_{pq} = d_{qp}$ is one of the minimal elements of the matrix D^j. Then, these lines and columns are replaced with a line and column that corresponds to the new cluster C_r and the dissimilarities between this new cluster and the previous clusters C_i, where $i \neq p, q$. The elements d_{rh}^{j+1} of the new line (and column) are obtained either as $\min\{d_{ph}^j, d_{qh}^j\}$, $\max\{d_{ph}^j, d_{qh}^j\}$, or $\frac{|C_p|}{|C_r|} d_{ph}^j + \frac{|C_q|}{|C_r|} d_{qh}^j$, for the single-link, complete-link, or group average methods, respectively. In any of these cases, it is not possible to obtain a value for d_{rh}^{j+1} that is less than the minimal value of an element of D^j. □

The last two methods captured by the Lance-Williams formula are the centroid method and the Ward method of clustering. As we observed before, Formula (7.3) shows that the dissimilarity of two clusters in the case of Ward's method equals the increase in the sum of the squared errors that results when the clusters are merged. The centroid method adopts the distance between the centroids as the distance between the corresponding clusters. Either method lacks the monotonicity properties.

Example 40. Let S be a synthetic data set that contains 35 points generated by using the function `setofpoints2` introduced in Example 25 as follows:

```
S1 <- setofpoints2(10,c(2,4),c(0.2,0.3))
S2 <- setofpoints2(15,c(3,3),c(0.3,0.3))
S3 <- setofpoints2(10,c(4,4),c(0.2,0.4))
S  <- rbind(S1,S2,S3)
colnames(S) <- c("x","y")
```

The plot of the objects is shown in Figure 7.6. Starting from the matrix S a `dist` object is produced by `d<-dist(S)`. Next, the function `hclust` is applied in order to produce the single-link hierarchical clustering `sLink`:

```
sLink <- hclust(d,method="single")
```

The dendrogram of the clustering is visualized using `plot(sLink)` and its representation is shown in Figure 7.7. To obtain three clusters, the dendrogram is "cut" at an appropriate level using the function call `rect.hclust(sLink,3)` generating the representation shown in Figure 7.8.

Similar clusterings are shown in Figures 7.9 and 7.10 obtained by the application of the complete-link and Ward methods, respectively.

Note that at the leaf-level, when clusters are sigletons, all methods produce exactly the same result. At higher levels the results diverge. The vertical axis shows the fusion level.

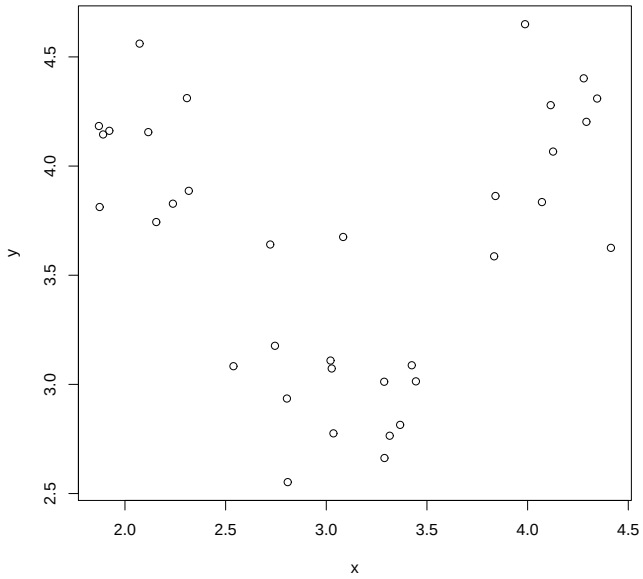

Fig. 7.6. Set containing 35 objects

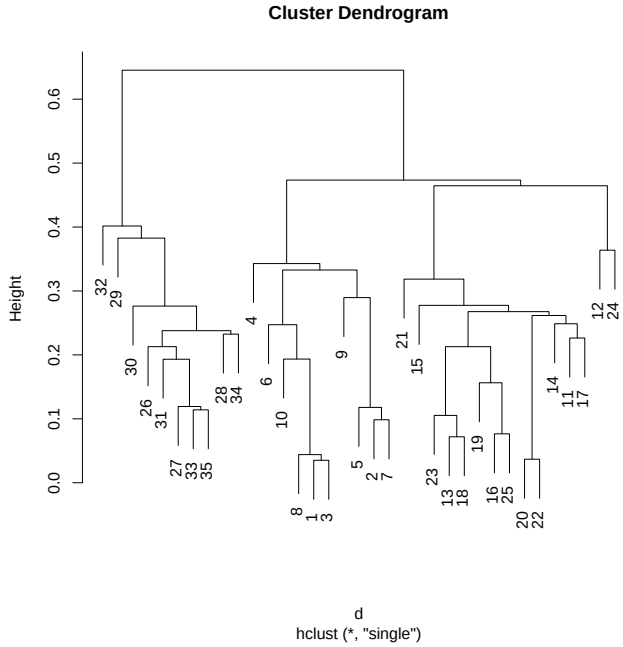

Fig. 7.7. Dendrogram of the single-link clustering
plot(

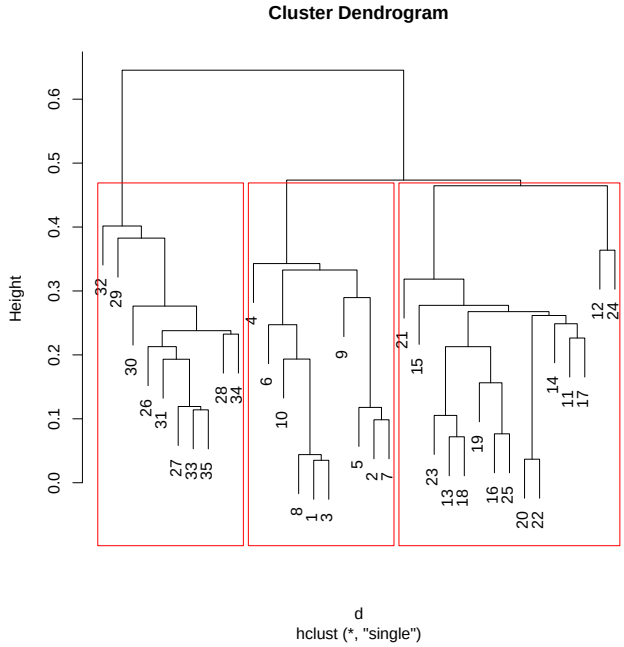

Fig. 7.8. Three clusters in S delimited by rectangles

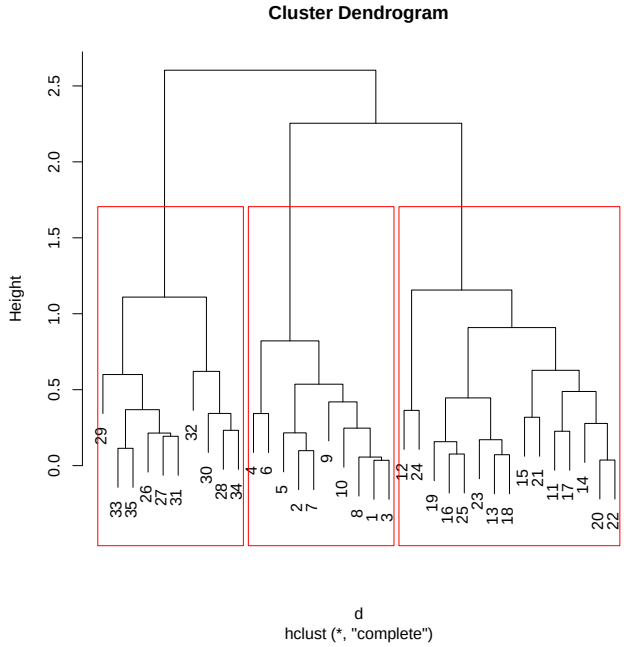

Fig. 7.9. Clusters delimited by rectangles obtained through the complete-link method

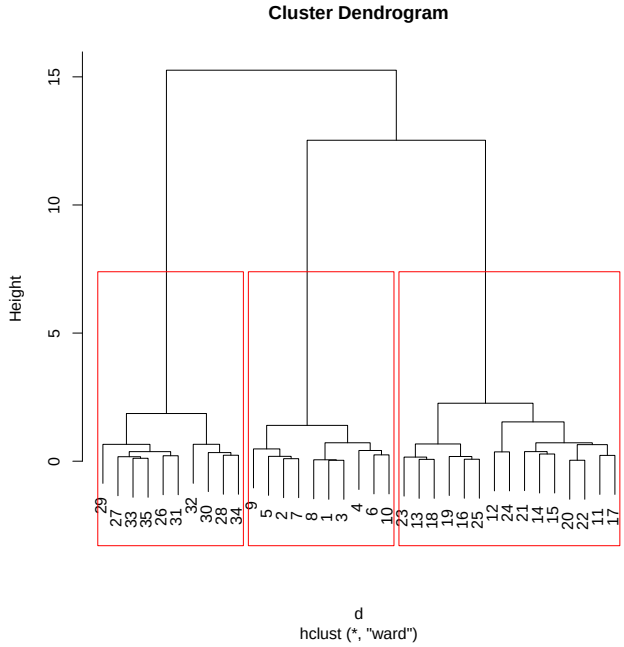

Fig. 7.10. Clusters in S delimited by rectangles obtained through the Ward method

8

Metric Multidimensional Scaling

Multidimensional scaling (MDS) is a process that allows us to represent a dissimilarity space using a low-dimensional Euclidean space. Scaling is important for visualizing the result of data explorations.

Two basic types of scaling algorithms exist. The *metric multidimensional scaling* starts with a finite dissimilarity space and produces a set of vectors that optimizes a function known as *strain*. The *non-metric multidimensional scaling* seeks a monotonic relationship between the values of a finite dissimilarity and the distances between the vectors that represent the elements of the dissimilarity space.

8.1 Metric Multidimensional Scaling

Metric multidimensional scaling (MMS) begins with a matrix of squares of Euclidean distances $D = (d_{ij}^2) \in \mathbb{R}^{m \times m}$ between m points situated in $\mathbb{R}^n$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ and seeks to determine these points starting from D. In general, $m \gg n$.

In view of the previous assumption, the rank of the matrix $X = (\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_m)$ is at most equal to n.

The definition of D implies that

$$d_{ij}^2 = \| \boldsymbol{x}_i - \boldsymbol{x}_j \|_2^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)'(\boldsymbol{x}_i - \boldsymbol{x}_j)$$

for $1 \leq i, j \leq m$. Clearly, this problem does not have a unique solution because the matrix D is the same for $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ and for $\boldsymbol{x}_1 + \boldsymbol{c}, \ldots, \boldsymbol{x}_m + \boldsymbol{c}$ for every $\boldsymbol{c} \in \mathbb{R}^n$.

The Gram matrix of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ is the matrix $G \in \mathbb{R}^{m \times m}$ given by $G = G_X = X'X$. Since $g_{pq} = \boldsymbol{x}_p' \boldsymbol{x}_q$ for $1 \leqslant p, q \leqslant m$, we have:

$$d_{ij}^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)'(\boldsymbol{x}_i - \boldsymbol{x}_j) = g_{ii} + g_{jj} - 2g_{ij} \tag{8.1}$$

for $1 \leq i, j \leq m$.

Suppose now that $F \in \mathbb{R}^{m\times m}$ is the Gram matrix of another sequence of vectors $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ such that $d(\boldsymbol{y}_i, \boldsymbol{y}_j) = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $1 \leqslant i, j \leqslant m$. Then, $g_{ii} + g_{jj} - 2g_{ij} = f_{ii} + f_{jj} - 2f_{ij}$ for $1 \leq i, j \leq m$.

Let $W = G - F$. Then, W is a symmetric matrix and $w_{ii} + w_{jj} - 2w_{ij} = 0$, so $w_{ij} = \frac{1}{2}(w_{ii} + w_{jj})$. Let

$$\boldsymbol{w} = \frac{1}{2}\begin{pmatrix} w_{11} \\ \vdots \\ w_{mm} \end{pmatrix}$$

and note that the matrix W can now be written as $W = \boldsymbol{w}\mathbf{1}_m' + \mathbf{1}_m\boldsymbol{w}'$, which proves that W is a special, rank-2 matrix. Consequently,

$$G = F + \boldsymbol{w}\mathbf{1}_m' + \mathbf{1}_m\boldsymbol{w}'. \tag{8.2}$$

Thus, the set of vectors that correspond to a distance matrix is not unique, and the Gram matrices of any two such sequences differ by a symmetric matrix of rank 2.

It is possible to construct $X = (\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_m)$ starting from D if we assume that the centroid of the vectors of X is $\mathbf{0}_n$, that is, if $\sum_{i=1}^m \boldsymbol{x}_i = \mathbf{0}_n$.

Let $A \in \mathbb{R}^{m\times m}$ be the matrix defined by $A = -\frac{1}{2}D$. Elementwise, this means that $a_{ij} = -\frac{1}{2}d_{ij}^2$ for $1 \leq i, j \leq m$. Consider the averages defined by:

$$\begin{aligned} a_{i\cdot} &= \frac{1}{m}\sum_{j=1}^m a_{ij}, \\ a_{\cdot j} &= \frac{1}{m}\sum_{i=1}^m a_{ij}, \\ a_{\cdot\cdot} &= \frac{1}{m^2}\sum_{i=1}^m\sum_{j=1}^m a_{ij}. \end{aligned}$$

The components of the Gram matrix $G \in \mathbb{C}^{m\times m}$, $g_{ij} = \boldsymbol{x}_i'\boldsymbol{x}_j$ for $1 \leq i, j \leq m$, can be expressed using these averages, assuming that the set of columns of X is centered in $\mathbf{0}_n$.

Theorem 33. *Let $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{n\times m}$ be a matrix such that $\sum_{i=1}^m \boldsymbol{x}_i = \mathbf{0}_n$ and let A be the matrix defined by $a_{ij} = -\frac{1}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel_2^2$ for $1 \leq i, j \leq m$. The components of the Gram matrix G, $g_{ij} = \boldsymbol{x}_i'\boldsymbol{x}_j$ are given by*

$$g_{ij} = a_{ij} - a_{i\cdot} - a_{\cdot j} + a_{\cdot\cdot}$$

for $1 \leq i, j \leq m$.

Proof. By Equality (8.1) we have

$$-2a_{ij} = g_{ii} + g_{jj} - 2g_{ij}$$

for $1 \leq i, j \leq m$. Note that

$$\sum_{i=1}^{m} g_{ij} = \sum_{j=1}^{m} g_{ij} = 0.$$

The averages introduced earlier can be written as

$$\begin{aligned}
-2a_{\cdot j} &= \frac{1}{m}\sum_{i=1}^{m} d_{ij}^2 = \frac{1}{m}\sum_{i=1}^{m}(g_{ii} + g_{jj} - 2g_{ij}) \\
&= \frac{1}{m}\sum_{i=1}^{m} g_{ii} + g_{jj} - 2\left(\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i'\right)\boldsymbol{x}_j \\
&= g_{jj} + \frac{1}{m}\sum_{i=1}^{m} g_{ii},
\end{aligned}$$

because $\sum_{i=1}^{m}\boldsymbol{x}_i = \mathbf{0}_n$. Similarly, we have

$$-2a_{i\cdot} = g_{ii} + \frac{1}{m}\sum_{j=1}^{m} g_{jj}. \tag{8.3}$$

Therefore,

$$\frac{1}{n^2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij}^2 = \frac{2}{m}\sum_{i=1}^{m} g_{ii}. \tag{8.4}$$

Thus, we have

$$\begin{aligned}
g_{ii} &= \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 - \frac{1}{m}\sum_{j=1}^{m} g_{jj} \\
g_{jj} &= \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 - \frac{1}{m}\sum_{i=1}^{m} g_{ii}
\end{aligned}$$

Equality (8.1) yields

$$\begin{aligned}
g_{ij} &= -\frac{1}{2}\left(d_{ij}^2 - g_{ii} - g_{jj}\right) \\
&= -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 + \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{x}_j\boldsymbol{x}_j' - \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 + \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i\boldsymbol{x}_i'\right) \\
&= -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 - \frac{1}{m}\sum_{j=1}^{m} d_{ij}^2 + \frac{1}{r^2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij}^2\right) \\
&= a_{ij} - a_{i\cdot} - a_{\cdot j} + a_{\cdot\cdot},
\end{aligned}$$

which completes the proof.

Corollary 4. *The Gram matrix $G = X'X$ of the sequence of $\mathbb{R}^n$ vectors $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ can be obtained from the matrix $A = \left(-\frac{1}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel_2^2\right)$ as $G = H_m A H_m$, where H_m is the centering matrix $H_m = I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}'_m$.*

Proof. The matrix $H_m A H_m$ can be written as

$$\begin{aligned} H_m A H_m &= \left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}'_m\right) A \left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}'_m\right) \\ &= \left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}'_m\right)\left(A - \frac{1}{m}A\mathbf{1}_m\mathbf{1}'_m\right) \\ &= A - \mathbf{1}_m\left(\frac{1}{m}\mathbf{1}'_m A\right) - \left(\frac{1}{m}A\mathbf{1}_m\right)\mathbf{1}' + \frac{1}{m^2}\mathbf{1}_m\left(\mathbf{1}'_m A\mathbf{1}_m\right)\mathbf{1}'_m. \end{aligned}$$

The terms of the above sum correspond to $a_{ij}, a_{\cdot j}, a_{i\cdot}$ and $a_{\cdot\cdot}$, respectively. The desired conclusion then follows from Theorem 33.

The rank of the matrix $G = X'X \in \mathbb{R}^{m\times m}$ is equal to the rank of X, namely $\mathsf{rank}(G) = n$. Since G is symmetric, positive semi-definite and of rank n, it follows that G has n non-negative eigenvalues and $m-n$ zero eigenvalues. By the Spectral Theorem for Hermitian matrices we have $G = U'DU$, where U is an orthogonal matrix, $U = (\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_m)$, $D = (\lambda_1, \ldots, \lambda_n, 0, \ldots, 0)$ and $\lambda_1 \geq \cdots \geq \lambda_n > 0$.

Taking into account that the last $m-n$ elements of the diagonal of G are 0 we can write $G = V'\mathsf{diag}(\lambda_1, \ldots, \lambda_n)V$, where $V \in \mathbb{R}^{n\times m}$. By defining X as $X = \mathsf{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})V \in \mathbb{R}^{n\times m}$ we have $G = X'X$ and the m columns of X yield the desired vectors in $\mathbb{R}^n$.

A more general problem begins with a matrix of dissimilarities $\Delta = (\delta_{ij}) \in \mathbb{R}^{m\times m}$ and seeks to determine whether there exists a sequence of vectors $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ in $\mathbb{R}^n$ such that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_{ij}$ for $1 \leq i, j \leq m$.

Lemma 3. *Let $A, G \in \mathbb{R}^{m\times m}$ be two matrices such that $G = H_m A H_m$, where H_m is the centering matrix $H_m = I_m - \frac{1}{n}\mathbf{1}_m\mathbf{1}'_m$. Then, $g_{ii} + g_{jj} - 2g_{ij} = a_{ii} + a_{jj} - 2a_{ij}$ for $1 \leq i, j \leq m$.*

Proof. We saw that if $G = H_m A H_m$, then $g_{ij} = a_{ij} - a_{i\cdot} - a_{\cdot j} + a_{\cdot\cdot}$. Therefore, we have

$$\begin{aligned} g_{ii} &= a_{ii} - 2a_{i\cdot} + a_{\cdot\cdot}, \\ g_{jj} &= a_{jj} - 2a_{\cdot j} + a_{\cdot\cdot}. \end{aligned}$$

This allows us to write

$$\begin{aligned} g_{ii} + g_{jj} - 2g_{ij} &= a_{ii} - 2a_{i\cdot} + a_{\cdot\cdot} + a_{jj} - 2a_{\cdot j} + a_{\cdot\cdot} \\ &\quad -2(a_{ij} - a_{i\cdot} - a_{\cdot j} + a_{\cdot\cdot}) \\ &= a_{ii} + a_{jj} - 2a_{ij}. \end{aligned}$$

Theorem 34. *Let $\Delta \in \mathbb{R}^{m\times m}$ be a matrix of dissimilarities, $A \in \mathbb{R}^{m\times m}$ be the matrix defined by $a_{ij} = -\frac{1}{2}\delta_{ij}^2$ for $1 \leq i,j \leq m$, and let G be the centered matrix $G = H_m A H_m$. If G is a positive semi-definite matrix and $\mathsf{rank}(G) = n$, then there exists a sequence $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ of vectors in $\mathbb{R}^n$ such that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_{ij}$ for $1 \leq i,j \leq m$.*

Proof. Since G is a symmetric, positive semi-definite matrix having rank n, it is possible to write $G = V'DV$, where $V = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_m) \in \mathbb{R}^{n\times m}$, $D = (\lambda_1, \ldots, \lambda_n)$ and $\lambda_1 \geq \cdots \geq \lambda_n > 0$.

Let $X = \mathsf{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})V \in \mathbb{R}^{n\times m}$. We claim that the distances between $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ equal the prescribed dissimilarities. Indeed, since $\boldsymbol{x}_i = \sqrt{\lambda_i}\boldsymbol{v}_i$, we have

$$\begin{aligned} d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 &= (\boldsymbol{x}_i - \boldsymbol{x}_j)'(\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= \boldsymbol{x}_i'\boldsymbol{x}_i + \boldsymbol{x}_j'\boldsymbol{x}_j - 2\boldsymbol{x}_i'\boldsymbol{x}_j \\ &= \lambda_i \boldsymbol{v}_i'\boldsymbol{v}_i + \lambda_j \boldsymbol{v}_j'\boldsymbol{v}_j - 2\lambda_i\lambda_j \boldsymbol{v}_i'\boldsymbol{v}_j \\ &= g_{ii} + g_{jj} - 2g_{ij} \\ &\quad \text{(by Lemma 3)} \\ &= a_{ii} + a_{jj} - 2a_{ij} = -2a_{ij} = \delta_{ij}^2, \end{aligned}$$

which is the desired conclusion.

Since $G = H_m A H_m$ and H_m has an eigenvalue equal to 0 it is clear that G also has such an eigenvalue. Therefore, $\mathsf{rank}(G) \leq m-1$, so there exist m vectors of dimensionality not larger than $m-1$ such that their distances are equal to the given dissimilarities.

We saw that the matrices XX' and $G = X'X$ have the same rank and their non-zero eigenvalues are positive numbers and have the same algebraic multiplicities for both matrices.

Let $\boldsymbol{w}$ be a principal component of the matrix $X' \in \mathbb{R}^{n\times m}$, that is, an eigenvector of the matrix XX'. Suppose that $\mathsf{rank}(G) = r$ and let $X' = UDV'$ be the thin SVD decomposition of the matrix X', where $D = (\sigma_1, \ldots, \sigma_r)$ and $U, V \in \mathbb{R}^{m\times r}$. The matrices U and V have orthogonal columns, so $U'U = V'V = I_r$. Since the numbers $\sigma_1, \ldots, \sigma_r$ are positive, D is invertible and we obtain $U = X'VD^{-1}$. Thus, MDS involves a process that is dual to the usual PCA; some authors refer to it as the *dual PCA*.

R deals with metric MDS using the function `cmdscale`.

Example 41. The function call

```
bid <- cmdscale(mdist,eig=TRUE,k=2)
```

is applied to a matrix `mdist` that reflects *driving distances* between five US northeastern cities: Boston, Providence, Harford, New York and Concord (NH). This matrix is

```
> mdist
       [,1]   [,2]   [,3]   [,4]   [,5]
[1,]   0.00  41.90  92.88 189.90  63.47
[2,]  41.90   0.00  65.36 154.84  95.78
[3,]  92.88  65.36   0.00  99.76 115.59
[4,] 189.90 154.84  99.76   0.00 213.78
[5,]  63.27  95.78 115.59 213.78   0.00
```

The object `bid` has the following structure:

```
> bid
$points
          [,1]       [,2]
[1,] -58.16771  20.378775
[2,] -19.31546  34.301551
[3,]  29.85819  -8.803585
[4,] 129.61745  -7.808648
[5,] -82.00508 -38.082192

$eig
[1] 28173.596590  3180.622307    32.655384    -1.526754    -6.099726

$x
NULL

$ac
[1] 0

$GOF
[1] 0.9987169 0.9989596
```

The points components are the coordinates of five points in two dimensions, that are placed on a plot using

```
> x <- bid$points[,1]
> y <- bid$points[,2]
> plot(x,y,xlab="First Coord",ylab="Second Coord",main="Metric MDS",type="n")
> text(x,y,labels=c("Boston","Providence","Hartford","New York","Concord"),cex = 0.7)
```

which results in the representation contained in Figure 8.1.

Note that by rotationg this plot by 90 degrees clockwise we obtain an image that is closer to the real geographic position of these cities, as shown in Figure 8.2. Of course, the representation is approximative, but the relative positions of the cities is reasonably close to their real placement.

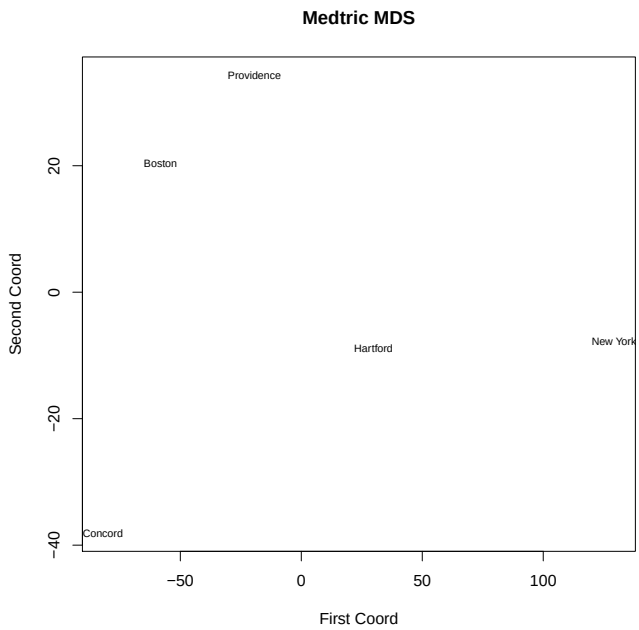

Fig. 8.1. Representation of the five cities.

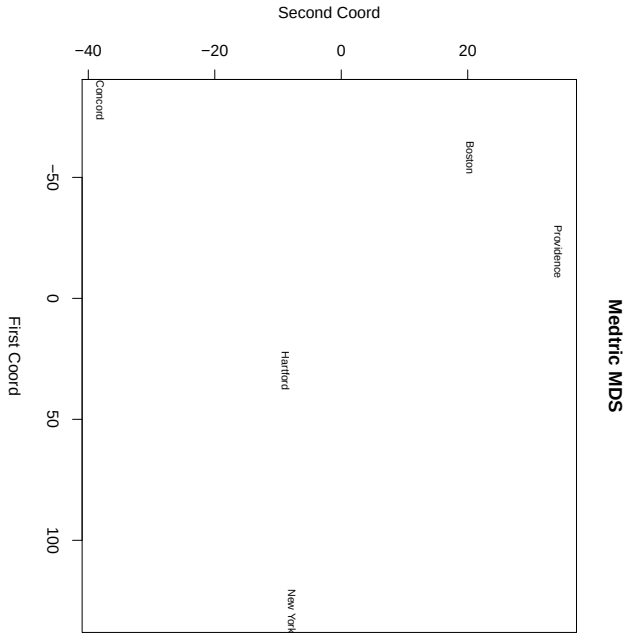

Fig. 8.2. Representation of the five cities.