Cluster Analysis

Contents

- Required packages

```
knitr::opts_chunk$set(echo = TRUE)
#install.packages("dplyr")
#install.packages("stargazer")
#install.packages("ade4")
#install.packages("magrittr")
#install.packages("cluster")
#install.packages("factoextra")
install.packages("cluster.datasets")
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/4.0'
## (as 'lib' is unspecified)
```

Definition of a distance

- A distance function or a metric on $\mathbb{R}^n$, $n \geq 1$, is a function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.
- A distance function must satisfy some required properties or axioms.
- There are three main axioms.
- A1. $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$ (identity of indiscernibles);
- A2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (symmetry);
- A3. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ (triangle inequality), where $\mathbf{x} = (x_1, \cdots, x_n)$, $\mathbf{y} = (y_1, \cdots, y_n)$ and $\mathbf{z} = (z_1, \cdots, z_n)$ are all vectors of $\mathbb{R}^n$.
- We should use the term *dissimilarity* rather than *distance* when not all the three axioms A1-A3 are valid.
- Most of the time, we shall use, with some abuse of vocabulary, the term distance.

Exercice 1

- Prove that the three axioms A1-A3 imply the non-negativity condition:

$$d(\mathbf{x}, \mathbf{y}) \geq 0.$$

Euclidean distance

- It is defined by:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

- A1-A2 are obvious.
- The proof of A3 is provided below.

Exercice 2

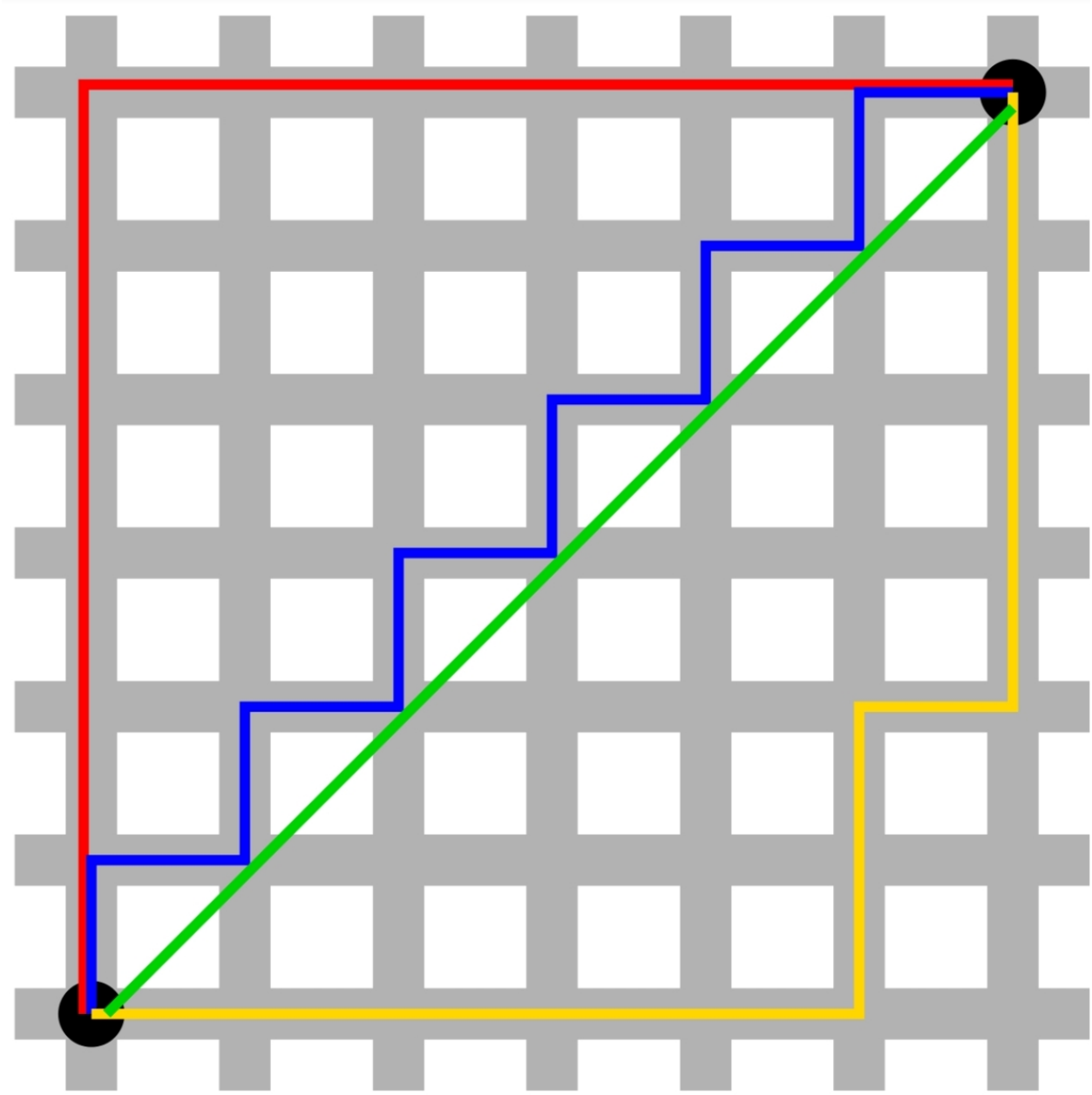
- Is the squared Euclidian distance a true distance?

Manhattan distance

- The Manhattan distance also called taxi-cab metric or city-block metric is defined by:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

- A1-A2 hold.
- A3 also holds using the fact that $|a + b| \leq |a| + |b|$ for any reals a, b.
- There exists also a weighted version of the Manhattan distance called the Canberra distance.



```
x = c(0, 0)
y = c(6,6)
dist(rbind(x, y), method = "euclidian")
```

```
##          x
## y 8.485281
```

```
dist(rbind(x, y), method = "euclidian",diag=T,upper=T)
```

```
##          x        y
## x 0.000000 8.485281
## y 8.485281 0.000000
```

```
6*sqrt(2)
```

```
## [1] 8.485281
```

```
dist(rbind(x, y), method = "manhattan")
```

```
##    x
## y 12
```

```
dist(rbind(x, y), method = "manhattan",diag=T,upper=T)
```

```
##    x  y
## x  0 12
## y 12  0
```

Canberra distance

- It is defined by:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}.$$

- Note that the term $|x_i - y_i|/(|x_i| + |y_i|)$ is not properly defined as: $x_i = y_i = 0$.
- By convention we set that term to be zero in that case.
- The Canberra distance is specially sensitive to small changes near zero.

```
x = c(0, 0)
y = c(6,6)
dist(rbind(x, y), method = "canberra")
```

```
##   x
## y 2
```

```
6/6+6/6
```

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

Exercice 3

- Prove that the Canberra distance is a true distance, i.e. that it satisfies A1-A3.

Minkowski distance

- Both the Euclidian and the Manattan distances are special cases of the Minkowski distance which is defined, for $p \geq 1$, by:

$$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1} |x_i - y_i|^p \right]^{1/p}.$$

- For $p = 1$, we get the Manhattan distance.
- For $p = 2$, we get the Euclidian distance.
- Let us also define:

$$\|\mathbf{x}\|_p \equiv \left[\sum_{i=1}^{n} |x_i|^p \right]^{1/p},$$

where $\|\cdot\|_p$ is known as the p-norm or Minkowski norm.

- Note that the Minkowski distance and norm are related by:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p.$$

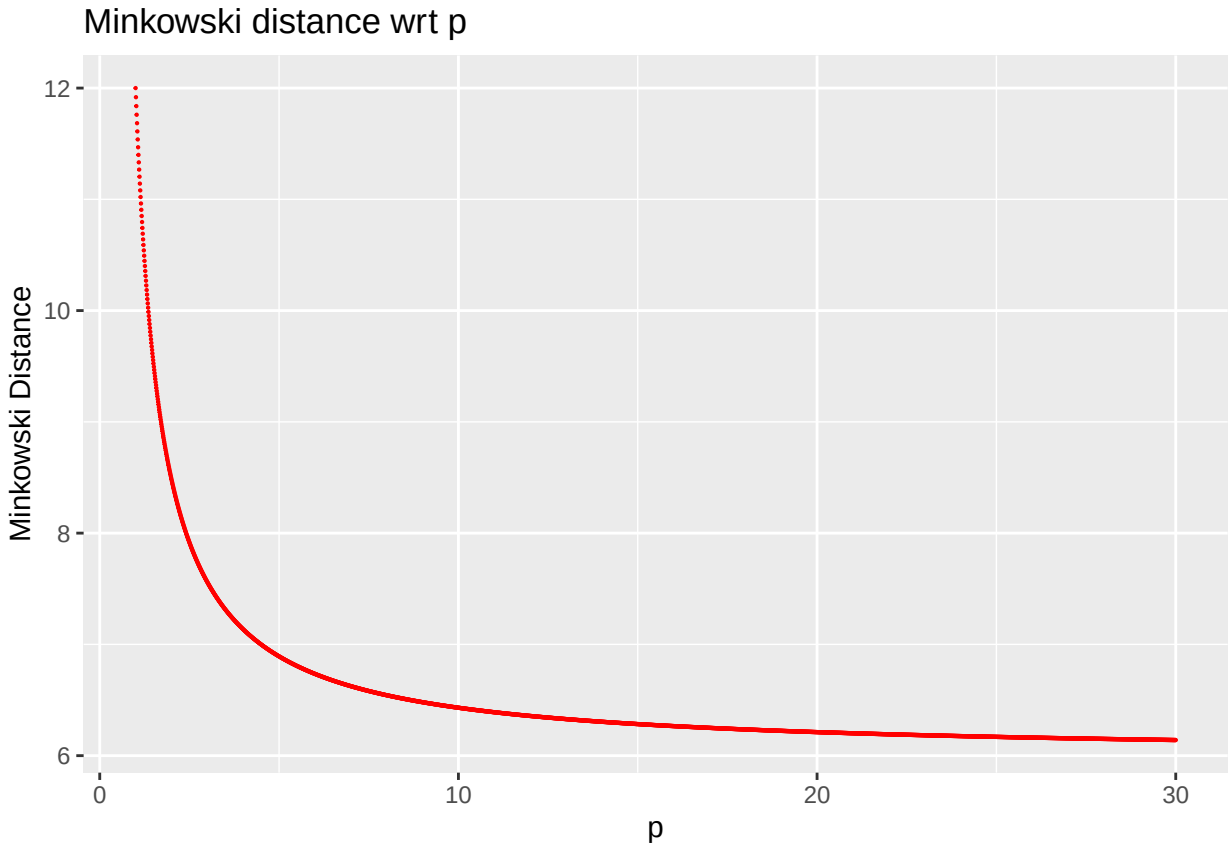
- Conversely, we have:

$$\|\mathbf{x}\|_p = d(\mathbf{x}, \mathbf{0}),$$

where $\mathbf{0}$ is the null-vetor of $\mathbb{R}^n$.

```
library("ggplot2")
x = c(0, 0)
y = c(6,6)
MinkowDist=c() # Initialiser à vide la liste
for (p in seq(1,30,.01))
{
MinkowDist=c(MinkowDist,dist(rbind(x, y), method = "minkowski", p = p))
}

ggplot(data =data.frame(x = seq(1,30,.01), y=MinkowDist ) , mapping = aes( x=x, y= y))+
  geom_point(size=.1,color="red")+
  xlab("p")+ylab("Minkowski Distance")+ggtitle("Minkowski distance wrt p")
```



Exercice 4

Produce a similar graph using "The Economist" theme. Indicate on the graph the Manhattan, the Euclidian distances as well as the Chebyshev distance introduces below.

Chebyshev distance

- At the limit, we get the Chebyshev distance which is defined by:

$$d(\mathbf{x},\mathbf{y}) = \max_{i=1,\cdots,n}(|x_i - y_i|) = \lim_{p\to\infty}\left[\sum_{i=1}|x_i - y_i|^p\right]^{1/p}.$$

- The corresponding norm is:

$$\|\mathbf{x}\|_\infty = \max_{i=1,\cdots,n}(|x_i|).$$

Minkowski inequality

- The proof of the triangular inequality A3 is based on the Minkowski inequality:
- For any nonnegative real numbers $a_1,\cdots,a_n$; $b_1,\cdots,b_n$, and for any $p \geq 1$, we have:

$$\left[\sum_{i=1}^n (a_i + b_i)^p\right]^{1/p} \leq \left[\sum_{i=1}^n a_i^p\right]^{1/p} + \left[\sum_{i=1}^n b_i^p\right]^{1/p}.$$

- To prove that the Minkowski distance satisfies A3, notice that

$$\sum_{i=1}^n |x_i - z_i|^p = \sum_{i=1}^n |(x_i - y_i) + (y_i - z_i)|^p.$$

- Since for any reals x, y, we have: $|x+y| \leq |x| + |y|$, and using the fact that x^p is increasing in $x \geq 0$, we obtain:

$$\sum_{i=1}^n |x_i - z_i|^p \leq \sum_{i=1}^n (|x_i - y_i| + |y_i - z_i|)^p.$$

- Applying the Minkowski inequality with $a_i = |x_i - y_i|$ and $b_i = |y_i - z_i|$, $i = 1,\cdots,n$, we get:

$$\sum_{i=1}^n |x_i - z_i|^p \leq \left(\sum_{i=1}^n |x_i - y_i|^p\right)^{1/p} + \left(\sum_{i=1}^n |y_i - z_i|^p\right)^{1/p}.$$

Exercice 5

To illustarate the Minkowski inequality, draw 100 times two lists of 100 draws from the lognormal distribution with mean 1600 and standard-deviation 300. Illustrate with a graph the gap between the two drawn lists.

Hölder inequality

- The proof of the Minkowski inequality itself requires the Hölder inequality:
- For any nonnegative real numbers $a_1, \cdots, a_n$; $b_1, \cdots, b_n$, and any $p, q > 1$ with $1/p + 1/q = 1$, we have:

$$\sum_{i=1}^{n} a_i b_i \leq \left[\sum_{i=1}^{n} a_i^p\right]^{1/p} \left[\sum_{i=1}^{n} b_i^q\right]^{1/q}$$

- The proof of the Hölder inequality relies on the Young inequality:
- For any $a, b > 0$, we have

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q},$$

  with equality occuring iff: $a^p = b^q$.
- To prove the Young inequality, one can use the (strict) convexity of the exponential function.
- For any reals x, y, we have:

$$e^{\frac{x}{p}+\frac{y}{q}} \leq \frac{e^x}{p} + \frac{e^y}{q}.$$

- We then set: $x = p \ln a$ and $y = q \ln b$ to get the Young inequality.
- A good reference on inequalities is: Z. Cvetkovski, Inequalities: theorems, techniques and selected problems, 2012, Springer Science & Business Media. # Cauchy-Schwartz inequality
- Note that the triangular inequality for the Minkowski distance implies:

$$\sum_{i=1}^{n} |x_i| \leq \left[\sum_{i=1}^{n} |x_i|^p\right]^{1/p}.$$

- Note that for $p = 2$, we have $q = 2$. The Hölder inequality implies for that special case

$$\sum_{i=1}^{n} |x_i y_i| \leq \sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}.$$

- Since the LHS od thes above inequality is greater then $|\sum_{i=1}^{n} x_i y_i|$, we get the Cauchy-Schwartz inequality

$$|\sum_{i=1}^{n} x_i y_i| \leq \sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}.$$

* Using the dot product notation called also scalar product noation: $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$, and the norm notation $\|\cdot\|_2\|$, the Cauchy-Schwart inequality is:

$$|\mathbf{x} \cdot \mathbf{y}| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2.$$

Pearson correlation distance

- The Pearson correlation coefficient is a similarity measure on $\mathbb{R}^n$ defined by:

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})(y_i - \bar{\mathbf{y}})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})^2 \sum_{i=1}^{n}(y_i - \bar{\mathbf{y}})^2}},$$

  where $\bar{\mathbf{x}}$ is the mean of the vector $\mathbf{x}$ defined by:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

- Note that the Pearson correlation coefficient satisfies P2 and is invariant to any positive linear transformation, i.e.:
$$\rho(\alpha\mathbf{x}, \mathbf{y}) = \rho(\mathbf{x}, \mathbf{y}),$$
for any $\alpha > 0$.
- The Pearson distance (or correlation distance) is defined by:
$$d(\mathbf{x}, \mathbf{y}) = 1 - \rho(\mathbf{x}, \mathbf{y}).$$
- Note that the Pearson distance does not satisfy A1 since $d(\mathbf{x}, \mathbf{x}) = 0$ for any non-zero vector $\mathbf{x}$. It neither satisfies the triangle inequality. However, the symmetry property is fullfilled.

Cosine correlation distance

- The cosine of the angle θ between two vectors $\mathbf{x}$ and $\mathbf{y}$ is a measure of similarity given by:
$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}}.$$
- Note that the cosine of the angle between the two centred vectors $\mathbf{x} - \bar{\mathbf{x}}\mathbf{1}$ and $\mathbf{y} - \bar{\mathbf{y}}\mathbf{1}$ coincides with the Pearson correlation coefficient of $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{1}$ is a vector of units of $\mathbb{R}^n$.
- The cosine correlation distance is defined by:
$$d(\mathbf{x}, \mathbf{y}) = 1 - \cos(\theta).$$
- It shares similar properties than the Pearson correlation distance. Likewise, Axioms A1 and A3 are not satisfied.

Spearman correlation distance

- To calculate the Spearman's rank-order correlation, we need to map seperately each of the vectors to ranked data values:
$$\mathbf{x} \to \text{rank}(\mathbf{x}) = (x_1^r, \cdots, x_n^r).$$
- Here, x_i^r is the rank of x_i among the set of values of $\mathbf{x}$.
- We illustrate this transformation with a simple example:
- If $\mathbf{x} = (3, 1, 4, 15, 92)$, then the rank-order vector is $\text{rank}(\mathbf{x}) = (2, 1, 3, 4, 5)$.

```
x=c(3, 1, 4, 15, 92)
rank(x)
```

```
## [1] 2 1 3 4 5
```

- The Spearman's rank correlation of two numerical variables $\mathbf{x}$ and $\mathbf{y}$ is simply the Pearson correlation of the two correspnding rank-order variables $\text{rank}(\mathbf{x})$ and $\text{rank}(\mathbf{y})$, i.e. $\rho(\text{rank}(\mathbf{x}), \text{rank}(\mathbf{y}))$. This measure is is useful because it is more robust against outliers than the Pearson correlation.
- If all the n ranks are distinct, it can be computed using the following formula:
$$\rho(\text{rank}(\mathbf{x}), \text{rank}(\mathbf{y})) = 1 - \frac{6\sum_{i=1}^n d_i^2}{n(n^2-1)},$$
where $d_i = x_i^r - y_i^r$, $i = 1, \cdots, n$.
- The spearman distance is then defined by:
$$d(\mathbf{x}, \mathbf{y}) = 1 - \rho(\text{rank}(\mathbf{x}), \text{rank}(\mathbf{y})).$$

- It can be shown that easaly that it is not a proper distance.
- If all the n ranks are distinct, we get:

$$d(\mathbf{x}, \mathbf{y}) = \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}.$$

```
x=c(3, 1, 4, 15, 92)
rank(x)
```

```
## [1] 2 1 3 4 5
```

```
y=c(30,2 , 9, 20, 48)
rank(y)
```

```
## [1] 4 1 2 3 5
```

```
d=rank(x)-rank(y)
d
```

```
## [1] -2  0  1  1  0
```

```
cor(rank(x),rank(y))
```

```
## [1] 0.7
```

```
1-6*sum(d^2)/(5*(5^2-1))
```

```
## [1] 0.7
```

Exercice 6

For the two vectors $\mathbf{x} = (22, 34, 1, 12, 25, 56, 7)$ and $\mathbf{y} = (2, 64, 12, 2, 22, 5, 8)$:

- Calculate the ranks for each vector.
- Deduce the Spearman correlation distance from that ranks.
- Deduce the Spearman correlation distance from the above dispalyed alternative equation.
- Calcualate the Spearman correlation distance using the R function.

Kendall tau distance

- The Kendall rank correlation coefficient is calculated from the number of correspondances between the rankings of $\mathbf{x}$ and the rankings of $\mathbf{y}$.
- The number of pairs of observations among n observations or values is:

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

- The pairs of observations (x_i, x_j) and (y_i, y_j) are said to be *concordant* if:

$$\text{sign}(x_j - x_i) = \text{sign}(y_j - y_i),$$

  and to be *discordant* if:

$$\text{sign}(x_j - x_i) = -\text{sign}(y_j - y_i),$$

  where sign$(\cdot)$ returns 1 for positive numbers and -1 negative numbers and 0 otherwise.
- If $x_i = x_j$ or $y_i = y_j$ (or both), there is a tie.

- The Kendall τ coefficient is defined by (neglecting ties):

$$\tau = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \text{sign}(x_j - x_i)\text{sign}(y_j - y_i).$$

- Let n_c (resp. n_d) be the number of concordant (resp. discordant) pairs, we have

$$\tau = \frac{2(n_c - n_d)}{n(n-1)}.$$

- The Kendall tau distance is then:

$$d(\mathbf{x}, \mathbf{y}) = 1 - \tau.$$

- Remark: the triangular inequality may fail in cases where there are ties.

```
x=c(3, 1, 4, 15, 92)
y=c(30,2 , 9, 20, 48)
tau=0
for (i in 1:5)
{
tau=tau+sign(x -x[i])%*%sign(y -y[i])
}
tau=tau/(5*4)
tau
```

```
##      [,1]
## [1,]  0.6
```

```
cor(x,y, method="kendall")
```

```
## [1] 0.6
```

Exercice 7

For the two vectors $\mathbf{x} = (22, 34, 1, 12, 25, 56, 7)$ and $\mathbf{y} = (2, 64, 12, 2, 22, 5, 8)$:

- List all pairs of coordinates.
- How many pairs are there?
- For each pair and each cector, compute the signs of the differences in coordinates.
- Deduce the Kendall tau coefficient using the above computations.
- Calcualte the Kendall tau coefficient using the R function.

Variables standardization

- Variables are often standardized before measuring dissimilarities.
- Standardization converts the original variables into uniteless variables.
- A well known method is the z-score transformation:

$$\mathbf{x} \to (\frac{x_1 - \bar{\mathbf{x}}}{s_{\mathbf{x}}}, \cdots, \frac{x_n - \bar{\mathbf{x}}}{s_{\mathbf{x}}}),$$

  where $s_{\mathbf{x}}$ is the sample standard deviation given by:

$$s_{\mathbf{x}} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{\mathbf{x}})^2.$$

- The transformed variable will have a mean of 0 and a variance of 1.
- The result obtained with Pearson correlation measures and standardized Euclidean distances are comparable.
- For other methods, see: Milligan, G. W., & Cooper, M. C. (1988). A study of standardization of variables in cluster analysis. *Journal of classification*, *5*(2), 181-204.

```
x=c(3, 1, 4, 15, 92)
y=c(30,2 , 9, 20, 48)
(x-mean(x))/sd(x)
```

```
## [1] -0.5134116 -0.5647527 -0.4877410 -0.2053646  1.7712699
```

```
scale(x)
```

```
##            [,1]
## [1,] -0.5134116
## [2,] -0.5647527
## [3,] -0.4877410
## [4,] -0.2053646
## [5,]  1.7712699
## attr(,"scaled:center")
## [1] 23
## attr(,"scaled:scale")
## [1] 38.9551
```

```
(y-mean(y))/sd(y)
```

```
## [1]  0.45263128 -1.09293895 -0.70654639 -0.09935809  1.44621214
```

```
scale(y)
```

```
##             [,1]
## [1,]  0.45263128
## [2,] -1.09293895
## [3,] -0.70654639
## [4,] -0.09935809
## [5,]  1.44621214
## attr(,"scaled:center")
## [1] 21.8
## attr(,"scaled:scale")
## [1] 18.11629
```

Similarity measures for binary data

- A common simple situation occurs when all information is of the presence/absence of 2-level qualitative characters.
- We assume there are n characters.
- *The presence of the character is coded by 1 and the absence by 0.
- We have have at our disposal two vectors.
- **x** is observed for a first individual (or object).
- **y** is observed for a second individual.
- We can then calculate the following four statistics:

$a = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$.

$b = \mathbf{x} \cdot (\mathbf{1} - \mathbf{y}) = \sum_{i=1}^{n} x_i (1 - y_i)$.

$c = (\mathbf{1} - \mathbf{x}) \cdot \mathbf{y} = \sum_{i=1}^{n} (1 - x_i) y_i$.

$d = (\mathbf{1} - \mathbf{x}) \cdot (\mathbf{1} - \mathbf{y}) = \sum_{i=1}^{n} (1 - x_i)(1 - y_i)$.

- The counts of matches are a for $(1, 1)$ and d for $(0, 0)$;
- The counts of mismatches are b for $(1, 0)$ and c for $(0, 1)$.
- Note that obviously: $a + b + c + d = n$.
- This gives a very useful 2×2 association table.

		Second individual		
		1	0	*Totals*
First individual	1	a	b	$a + b$
	0	c	d	$c + d$
Totals		$a + c$	$b + d$	n

Table 9 Binary Variables for Eight People

Person	Sex (Male = 1, Female = 0)	Married (Yes = 1, No = 0)	Fair Hair = 1, Dark Hair = 0	Blue Eyes = 1, Brown Eyes = 0	Wears Glasses (Yes = 1, No = 0)	Round Face = 1, Oval Face = 0	Pessimist = 1, Optimist = 0	Evening Type = 1, Morning Type = 0	Is an Only Child (Yes = 1, No = 0)	Left-Handed = 1, Right-Handed = 0
Ilan	1	0	1	1	0	0	1	0	0	0
Jacqueline	0	1	0	0	1	0	0	0	0	0
Kim	0	0	1	0	0	0	1	0	0	1
Lieve	0	1	0	0	0	0	0	1	1	0
Leon	1	1	0	0	1	1	0	1	1	0
Peter	1	1	0	0	1	0	1	1	0	0
Talia	0	0	0	1	0	1	0	0	0	0
Tina	0	0	0	1	0	1	0	0	0	0

Table from Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis *(Vol. 344). John Wiley & Sons*

- The data shows 8 people (individuals) and 10 binary variables:
- Sex, Married, Fair Hair, Blue Eyes, Wears Glasses, Round Face, Pessimist, Evening Type, Is an Only Child, Left-Handed.

```
data=c(
1,0,1,1,0,0,1,0,0,0,
0,1,0,0,1,0,0,0,0,0,
0,0,1,0,0,0,1,0,0,1,
0,1,0,0,0,0,0,1,1,0,
1,1,0,0,1,1,0,1,1,0,
1,1,0,0,1,0,1,1,0,0,
0,0,0,1,0,1,0,0,0,0,
0,0,0,1,0,1,0,0,0,0
)
data=data.frame(matrix(data, nrow=8,byrow=T))
row.names(data)=c("Ilan","Jacqueline","Kim","Lieve","Leon","Peter","Talia","Tina")
names(data)=c("Sex", "Married", "Fair Hair", "Blue Eyes", "Wears Glasses", "Round Face", "Pessimist", "
```

- We are comparing the records for Ilan with Talia.

```
x=data["Ilan",]
y=data["Talia",]
knitr::kable(table(x, y)[2:1,2:1],"pipe")
```

	1	0
1	1	3
0	1	5

- Therefore: $a = 1$, $b = 3$, $c = 1$, $d = 5$.
- Note that interchanging Ilan and Talia would permute b and c while leaving a and d unchanged.
- A good similarity or dissimilarity coefficient must treat b and c symmetrically.
- A similarity measure is denoted by: $s(\mathbf{x}, \mathbf{y})$.
- The corresponding distance is then defined as:

$$d(\mathbf{x}, \mathbf{y}) = 1 - s(\mathbf{x}, \mathbf{y}).$$

- Alternatively, we have:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{1 - s(\mathbf{x}, \mathbf{y})}.$$

- A list of some of the similarity measures $s(\mathbf{x}, \mathbf{y})$ that have been suggested for binary data is shown below.
- A more extensive list can be found in: Gower, J. C., & Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of classification*, *3*(1), 5-48.

Coefficient	$s(\mathbf{x}, \mathbf{y})$	$d(\mathbf{x}, \mathbf{y}) = 1 - s(\mathbf{x}, \mathbf{y})$
Simple matching	$\frac{a+d}{a+b+c+d}$	$\frac{b+c}{a+b+c+d}$
Jaccard	$\frac{a}{a+b+c}$	$\frac{b+c}{a+b+c}$
Rogers and Tanimoto (1960)	$\frac{a+d}{a+2(b+c)+d}$	$\frac{2(b+c)}{a+2(b+c)+d}$
Gower and Legendre (1986)	$\frac{2(a+d)}{2(a+d)+b+c}$	$\frac{b+c}{2(a+d)+b+c}]$
Gower and Legendre (1986)	$\frac{2a}{2a+b+c}$	$\frac{b+c}{2a+b+c}$

- To calculate these coefficients, we use the function: dist.binary().
- All the distances in this package are of type $d(\mathbf{x}.\mathbf{y}) = \sqrt{1 - s(\mathbf{x}.\mathbf{y})}$.

```
library(ade4)
a=1
b=3
c=1
d=5
dist.binary(data[c("Ilan","Talia"),],method=2)^2
```

```
  Ilan
```

Talia 0.4

```
1-(a+d )/(a+b+c+d)
```

[1] 0.4

```
dist.binary(data[c("Ilan","Talia"),],method=1)^2
```

```
  Ilan
```

Talia 0.8

```
1-a/(a+b+c)
```

[1] 0.8

```
dist.binary(data[c("Ilan","Talia"),],method=4)^2
```

```
       Ilan
```

Talia 0.5714286

```
1-(a+d )/(a+2*(b+c)+d)
```

[1] 0.5714286

```
# One Gower coefficient is missing
dist.binary(data[c("Ilan","Talia"),],method=5)^2
```

```
       Ilan
```

Talia 0.6666667

```
1-2*a/(2*a+b+c)
```

[1] 0.6666667

- The reason for such a large number of possible measures has to do with the apparent uncertainty as to how to deal with the count of zero-zero matches d.
- The measues embedding d are sometimes called symmetrical.
- The other measues are called assymmetrical.
- In some cases, of course, zero_zero matches are completely equivalent to one–one matches, and therefore should be included in the calculated similarity measure.
- An example is gender, where there is no preference as to which of the two categories should be coded zero or one.
- But in other cases the inclusion or otherwise of d is more problematic; for example, when the zero category corresponds to the genuine absence of some property, such as wings in a study of insects.

Exercice 8

- Prove that the distances based on the Simple Matching coefficient and the Jaccard coefficient satisfy A3.
- Prove that the distances proposed by Gower and Legendre (1986) do not satisfy A3.
- Hint: Proofs and counterexamples have to be adapted from in the paper:
- Gower, J. C., & Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of classification*, *3*(1), 5-48.

Nominal variables

- We previuosly studied above binary variables which can only take on two states coded as $0, 1$.
- We generalize this approach to nominal variables which may take on more than two states.
- Eye's color may have for example four states: blue, brown, green, grey .
- Le M be the number of states and code the outcomes as $1, \cdots, M$.
- We may choose $1 =$ blue, $2 =$ brown, $3 =$ green, and $4 =$ grey.
- These states are not ordered in any way
- One strategy would be creating a new binary variable for each of the M nominal states.
- Then to put it equal to 1 if the corresponding state occurs and to 0 otherwise.
- After that, one could resort to one of the dissimilarity coeffi- cients of the previous subsection.
- The most common way of measuring the similarity or dissimilarity between two objects through categorial variables is the simple matching approach.
- If $\mathbf{x}, \mathbf{y}$, are both n nominal records for two individuals,
- Let define the function:

$$\delta(x_i, y_i) \equiv \begin{cases} 0, & \text{if } x_i = y_i; \\ 1, & \text{if } x_i \neq y_i. \end{cases}$$

- Let N_{a+d} be the number of attributes of the two individuals on which the two records match:

$$N_{a+d} = \sum_{i=1}^{n} \delta(x_i, y_i).$$

- Let N_{b+c} be the number of attributes on which the two records do not match:

$$N_{b+c} = n - N_{a+d}.$$

- Let N_d be the number of attributes on which the two records match in a "not applicable" category:

$$N_d = \sum_{i=1}^{n} \delta(x_i, y_i).$$

- The distance corresponding to the simple matching approach is:

$$d(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} \delta(x_i, y_i)}{n}.$$

- Therefore:
$$d(\mathbf{x}, \mathbf{y}) = \frac{N_{a+d}}{N_{a+d} + N_{b+c}}.$$
- Note that simple matching has exactly the same meaning as in the preceding section.

Gower's dissimilarity

- Gower's coefficient is a dissimilarity measure specifically designed for handling mixed attribute types or variables.
- See: GOWER, John C. A general coefficient of similarity and some of its properties. *Biometrics*, 1971, p. 857-871.
- The coefficient is calculated as the weighted average of attribute contributions.
- Weights usually used only to indicate which attribute values could actually be compared meaningfully.
- The formula is:
$$d(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} w_i \delta(x_i, y_i)}{\sum_{i=1}^{n} w_i}.$$
- The wheight w_i is put equal to 1 when both measurements x_i and y_i are nonmissing,
- The number $\delta(x_i, y_i)$ is the contribution of the ith measure or variable to the dissimilarity measure.
- It the ith measure is nominal, we take
$$\delta(x_i, y_i) \equiv \begin{cases} 0, & \text{if } x_i = y_i; \\ 1, & \text{if } x_i \neq y_i. \end{cases}$$
- If the ith measure is interval-scaled, we take instead:
$$\delta(x_i, y_i) \equiv \frac{|x_i - y_i|}{R_i},$$
where R_i is the range of variable i over the available data.
- Consider the following data set:

	variable							
object	1	2	3	4	5	6	7	8
Begonia	0	1	1	4	3	15	25	15
Broom	1	0	0	2	1	3	150	50
Camellia	0	1	0	3	3	1	150	50
Dahlia	0	0	1	4	2	16	125	50
Forget-me-not	0	1	0	5	2	2	20	15
Fuchsia	0	1	0	4	3	12	50	40
Geranium	0	0	0	4	3	13	40	20
Gladiolus	0	0	1	2	2	7	100	15
Heather	1	1	0	3	1	4	25	15
Hydrangea	1	1	0	5	2	14	100	60
Iris	1	1	1	5	3	8	45	10
Lily	1	1	1	1	2	9	90	25
Lily-of-the-valley	1	1	0	1	2	6	20	10
Peony	1	1	1	4	2	11	80	30
Pink Carnation	1	0	0	3	2	10	40	20
Red Rose	1	0	0	4	2	18	200	60
Scotch Rose	1	0	0	2	2	17	150	60
Tulip	0	0	1	2	1	5	25	10

Table 1: Flower dataset.

Data

from: Struyf, A., Hubert, M., & Rousseeuw, P. (1997). Clustering in an object-oriented environment. Journal of Statistical Software*, 1(4), 1-30.*

- The dataset contains 18 flowers and 8 characteristics:

1. Winters: binary, indicates whether the plant may be left in the garden when it freezes.
2. Shadow: binary, shows whether the plant needs to stand in the shadow.
3. Tubers (Tubercule): asymmetric binary, distinguishes between plants with tubers and plants that grow in any other way.
4. Color: nominal, specifies the flower's color (1=white, 2=yellow, 3= pink, 4=red, 5= blue).
5. Soil: ordinal, indicates whether the plant grows in dry (1), normal (2), or wet (3) soil.
6. Preference: ordinal, someone's preference ranking, going from 1 to 18.
7. Height: interval scaled, the plant's height in centimeters.
8. Distance: interval scaled, the distance in centimeters that should be left between the plants.

- The dissimilarity between Begonia and Broom (Genêt) can be calculated as follows:

Begonia

Genêt

```
library(cluster)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data <-flower %>%
rename(Winters=V1,Shadow=V2,Tubers=V3,Color=V4,Soil=V5,Preference=V6,Height=V7,Distance=V8) %>%
mutate(Winters=recode(Winters,"1"="Yes","0"="No"),
      Shadow=recode(Shadow,"1"="Yes","0"="No"),
      Tubers=recode(Tubers,"1"="Yes","0"="No"),
      Color=recode(Color,"1"="white", "2"="yellow", "3"= "pink", "4"="red", "5"="blue"),
      Soil=recode(Soil,"1"="dry", "2"="normal", "3"= "wet")
      )
```

```
res=lapply(data,class)
res=as.data.frame(res)
```

```
res[1,] %>%
knitr::kable()
```

Winters	Shadow	Tubers	Color	Soil	Preference	Height	Distance
factor	factor	factor	factor	ordered	ordered	numeric	numeric

```
flower[1:2,]
```

```
##   V1 V2 V3 V4 V5 V6  V7 V8
## 1  0  1  1  4  3 15  25 15
## 2  1  0  0  2  1  3 150 50
```

```
max(data$Height)-min(data$Height)
```

```
## [1] 180
```

```
max(data$Distance)-min(data$Distance)
```

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

$$\frac{|1-0|+|0-1|+|0-1|+1+|1-3|/2+|3-15|/17+|150-25|/180+|50-15|/50}{8} \approx 0.8875408$$

`daisy` *Dissimilarity Matrix Calculation*

Description

Compute all the pairwise dissimilarities (distances) between observations in the data set. The original variables may be of mixed types. In that case, or whenever `metric = "gower"` is set, a generalization of Gower's formula is used, see 'Details' below.

Usage

```
daisy(x, metric = c("euclidean", "manhattan", "gower"),
      stand = FALSE, type = list(), weights = rep.int(1, p),
      warnBin = warnType, warnAsym = warnType, warnConst = warnType,
      warnType = TRUE)
```

```
library(cluster)
(abs(1-0)+abs(0-1)+abs(0-1)+1+abs(1-3)/2+abs(3-15)/17+abs(150-25)/180+abs(50-15)/50)/8
```

```
## [1] 0.8875408
```

```
daisy(data[,1:8],metric = "Gower")
```

```
## Warning in daisy(data[, 1:8], metric = "Gower"): with mixed variables, metric
## "gower" is used automatically
```

```
## Dissimilarities :
```

```
##           1         2         3         4         5         6         7
## 2  0.8875408
## 3  0.5272467 0.5147059
## 4  0.3517974 0.5504493 0.5651552
## 5  0.4115605 0.6226307 0.3726307 0.6383578
## 6  0.2269199 0.6606209 0.3003268 0.4189951 0.3443627
## 7  0.2876225 0.5999183 0.4896242 0.3435866 0.4197712 0.1892974
## 8  0.4234069 0.4641340 0.6038399 0.2960376 0.4673203 0.5714869 0.4107843
## 9  0.5808824 0.4316585 0.4463644 0.8076797 0.3306781 0.5136846 0.5890931
## 10 0.6094363 0.4531046 0.4678105 0.5570670 0.3812908 0.4119281 0.5865196
## 11 0.3278595 0.7096814 0.5993873 0.6518791 0.3864788 0.4828840 0.5652369
## 12 0.4267565 0.5857843 0.6004902 0.5132761 0.5000817 0.5248366 0.6391340
## 13 0.5196487 0.5248366 0.5395425 0.7464461 0.2919118 0.4524510 0.5278595
## 14 0.2926062 0.5949346 0.6096405 0.3680147 0.5203431 0.3656863 0.5049837
## 15 0.6221814 0.3903595 0.5300654 0.5531454 0.4602124 0.5091503 0.3345588
## 16 0.6935866 0.3575163 0.6222222 0.3417892 0.7301471 0.5107843 0.4353758
## 17 0.7765114 0.1904412 0.5801471 0.4247141 0.6880719 0.5937092 0.5183007
## 18 0.4610294 0.4515114 0.7162173 0.4378268 0.4755310 0.6438317 0.4692402
##           8         9        10        11        12        13        14
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  0.6366422
## 10 0.6639706 0.4256127
## 11 0.4955474 0.4308007 0.3948121
## 12 0.4216503 0.4194036 0.3812092 0.2636029
## 13 0.5754085 0.2181781 0.3643791 0.3445670 0.2331699
## 14 0.4558007 0.4396650 0.3609477 0.2838644 0.1591503 0.3784314
## 15 0.4512255 0.2545343 0.4210784 0.4806781 0.4295752 0.3183007 0.4351307
## 16 0.6378268 0.6494690 0.3488562 0.7436683 0.6050654 0.5882353 0.4598039
## 17 0.4707516 0.6073938 0.3067810 0.7015931 0.5629902 0.5461601 0.5427288
## 18 0.1417892 0.5198529 0.8057598 0.5359477 0.5495507 0.5733252 0.5698121
##          15        16        17
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16 0.3949346
## 17 0.3528595 0.1670752
## 18 0.5096814 0.7796160 0.6125408
```

```
##
## Metric :  mixed ;  Types = N, N, N, N, O, O, I, I
## Number of objects : 18
```

USArrests

From datasets v3.6.2
by R-core R-core@R-project.org

99.99th Percentile

Violent Crime Rates By US State

This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

Keywords datasets

Usage

```
USArrests
```

Note

`USArrests` contains the data as in McNeil's monograph. For the `UrbanPop` percentages, a review of the table (No. 21) in the Statistical Abstracts 1975 reveals a transcription error for Maryland (and that McNeil used the same "round to even" rule that R's `round()` uses), as found by Daniel S Coven (Arizona).

See the example below on how to correct the error and improve accuracy for the '<n>.5' percentages.

Format

A data frame with 50 observations on 4 variables.

[,1]	Murder	numeric	Murder arrests (per 100,000)
[,2]	Assault	numeric	Assault arrests (per 100,000)
[,3]	UrbanPop	numeric	Percent urban population

References

McNeil, D. R. (1977) *Interactive Data Analysis*. New York: Wiley.

- We use a subset of the data by taking 15 random rows among the 50 rows in the data set.
- We are using the function sample().
- We standardize the data using the function scale().

```
USArrests
```

```
##                Murder Assault UrbanPop Rape
## Alabama          13.2     236       58 21.2
## Alaska           10.0     263       48 44.5
## Arizona           8.1     294       80 31.0
## Arkansas          8.8     190       50 19.5
## California        9.0     276       91 40.6
## Colorado          7.9     204       78 38.7
## Connecticut       3.3     110       77 11.1
## Delaware          5.9     238       72 15.8
## Florida          15.4     335       80 31.9
## Georgia          17.4     211       60 25.8
## Hawaii            5.3      46       83 20.2
## Idaho             2.6     120       54 14.2
## Illinois         10.4     249       83 24.0
## Indiana           7.2     113       65 21.0
## Iowa              2.2      56       57 11.3
## Kansas            6.0     115       66 18.0
## Kentucky          9.7     109       52 16.3
## Louisiana        15.4     249       66 22.2
## Maine             2.1      83       51  7.8
## Maryland         11.3     300       67 27.8
## Massachusetts     4.4     149       85 16.3
## Michigan         12.1     255       74 35.1
## Minnesota         2.7      72       66 14.9
## Mississippi      16.1     259       44 17.1
## Missouri          9.0     178       70 28.2
## Montana           6.0     109       53 16.4
## Nebraska          4.3     102       62 16.5
## Nevada           12.2     252       81 46.0
## New Hampshire     2.1      57       56  9.5
## New Jersey        7.4     159       89 18.8
## New Mexico       11.4     285       70 32.1
## New York         11.1     254       86 26.1
## North Carolina   13.0     337       45 16.1
## North Dakota      0.8      45       44  7.3
## Ohio              7.3     120       75 21.4
## Oklahoma          6.6     151       68 20.0
## Oregon            4.9     159       67 29.3
## Pennsylvania      6.3     106       72 14.9
## Rhode Island      3.4     174       87  8.3
## South Carolina   14.4     279       48 22.5
## South Dakota      3.8      86       45 12.8
## Tennessee        13.2     188       59 26.9
## Texas            12.7     201       80 25.5
## Utah              3.2     120       80 22.9
## Vermont           2.2      48       32 11.2
## Virginia          8.5     156       63 20.7
## Washington        4.0     145       73 26.2
## West Virginia     5.7      81       39  9.3
```

```
## Wisconsin         2.6      53       66 10.8
## Wyoming           6.8     161       60 15.6
```

```
set.seed(123)
ss <- sample(1:50,15)
df <- USArrests[ss, ]
df.scaled <- scale(df)
df.scaled
```

```
##                    Murder     Assault    UrbanPop        Rape
## New Mexico      0.58508090  1.02300309  0.22505574  0.61101857
## Iowa           -1.70220419 -1.54760088 -0.68923319 -1.43885018
## Indiana        -0.45911447 -0.90775622 -0.12659385 -0.48290177
## Arizona        -0.23535832  1.12403120  0.92835492  0.50261205
## Tennessee       1.03259320 -0.06585536 -0.54857336  0.09855138
## Texas           0.90828422  0.08007413  0.92835492 -0.03942055
## Oregon         -1.03093574 -0.39139036  0.01406598  0.33507470
## West Virginia  -0.83204139 -1.26696726 -1.95517172 -1.63595295
## Missouri       -0.01160217 -0.17810880  0.22505574  0.22666818
## Montana        -0.75745600 -0.95265760 -0.97055287 -0.93623813
## Nebraska       -1.18010651 -1.03123501 -0.33758361 -0.92638299
## California     -0.01160217  0.92197499  1.70198401  1.44870532
## South Carolina  1.33093473  0.95565102 -1.32220246 -0.33507470
## Nevada          0.78397525  0.65256671  0.99868483  1.98088278
## Florida         1.57955267  1.58427034  0.92835492  0.59130829
## attr(,"scaled:center")
##     Murder    Assault   UrbanPop       Rape
##   9.046667 193.866667  66.800000  25.900000
## attr(,"scaled:scale")
##    Murder   Assault  UrbanPop      Rape
##  4.022236 89.084123 14.218700 10.146991
```

- The R functions for computing distances.

1. dist() function accepts only numeric data.
2. get_dist() function [factoextra package] accepts only numeric data. it supports correlation-based distance measures.
3. daisy() function [cluster package] is able to handle other variable types (nominal, ordinal, ...).

- Remark: All these functions compute distances between rows of the data.
- Remark: If we want to compute pairwise distances between variables, we must transpe the data to have variables in the rows.
- We first compute Euclidian distances

```
dist.eucl <- dist(df.scaled, method = "euclidean")

round(as.matrix(dist.eucl)[1:3, 1:3], 1)
```

```
##            New Mexico Iowa Indiana
## New Mexico        0.0  4.1     2.5
## Iowa              4.1  0.0     1.8
## Indiana           2.5  1.8     0.0
```

```
round(sqrt(sum((df.scaled["New Mexico",]-df.scaled["Iowa",])^2)),1)
```

```
## [1] 4.1
```

```
round(sqrt(sum((df.scaled["New Mexico",]-df.scaled["Indiana",])^2)),1)
```

```
## [1] 2.5
```

```
round(sqrt(sum((df.scaled["Iowa",]
-df.scaled["Indiana",])^2)),1)
```

```
## [1] 1.8
```

- We also compute correlation based distances.

```
library("factoextra")
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
dist.cor <- get_dist(df.scaled, method = "pearson")
round(as.matrix(dist.cor)[1:3, 1:3], 1)
```

```
##            New Mexico Iowa Indiana
## New Mexico        0.0  1.7     2.0
## Iowa              1.7  0.0     0.3
## Indiana           2.0  0.3     0.0
```

```
round(1-cor(df.scaled["New Mexico",],df.scaled["Iowa",]),1)
```

```
## [1] 1.7
```

```
round(1-cor(df.scaled["New Mexico",],df.scaled["Indiana",]),1)
```

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

```
round(1-cor(df.scaled["Iowa",],df.scaled["Indiana",]),1)
```

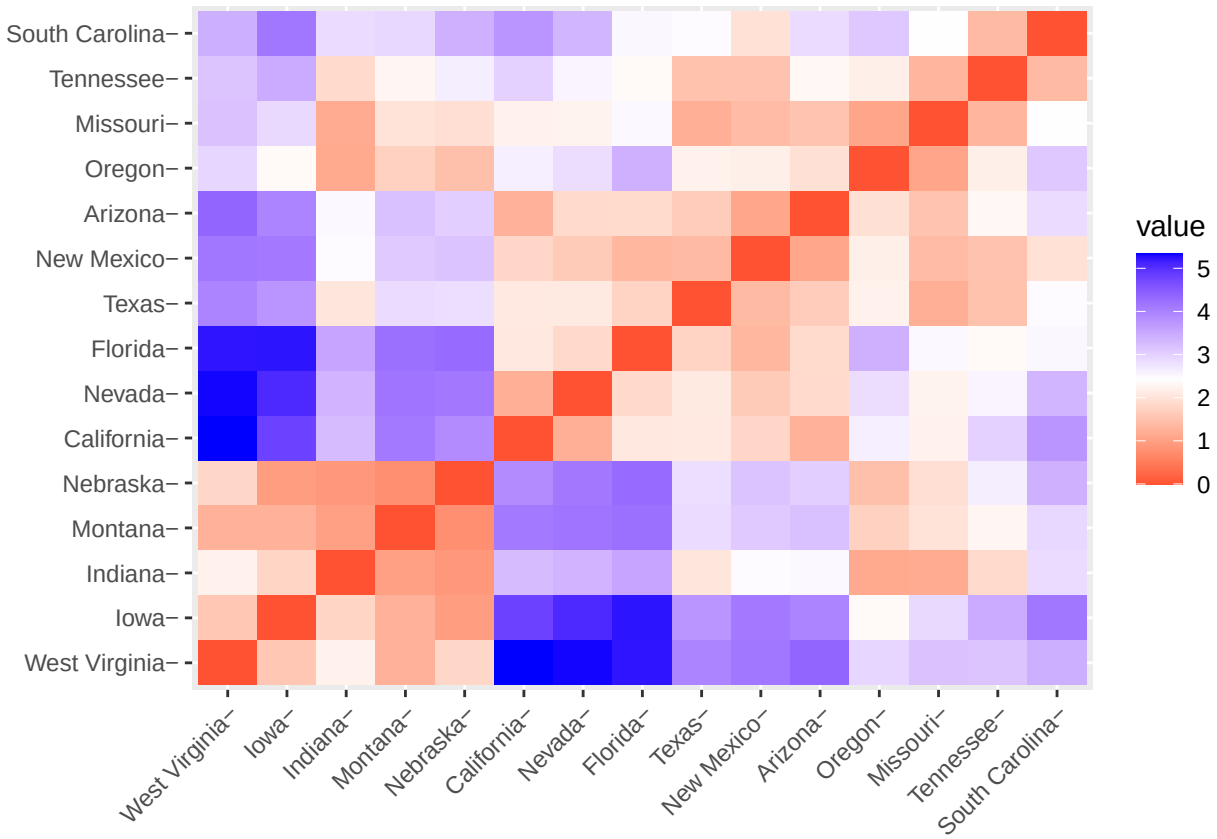
```
## [1] 0.3
```

Visualizing distance matrices

- A simple solution for visualizing the distance matrices is to use the function fviz_dist() [factoextra package].

- Other specialized methods will be described later on.

```
library(factoextra)
fviz_dist(dist.eucl)
```



Partitioning Clustering

- Partitioning clustering are clustering methods used to classify observations within a data set, into multiple groups based on their similarity.
- The algorithms require the analyst to specify the number of clusters to be generated.
- This chapter describes the commonly used partitioning clustering, including:

1. K-means clustering (MacQueen, 1967), in which, each cluster is represented by the center or means of the data points belonging to the cluster. The K-means method is sensitive to anomalous data points and outliers.
2. K-medoids clustering or PAM (Partitioning Around Medoids, Kaufman & Rousseeuw, 1990), in which, each cluster is represented by one of the objects in the cluster. PAM is less sensitive to outliers compared to k-means.
3. CLARA algorithm (Clustering Large Applications), which is an extension to PAM adapted for large data sets.

K-Means Clustering

- The description of the algorithm is based on:
- HARTIGAN, John A. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Table 4.1 Nutrients in Meat, Fish, and

[*The Yearbook of Agriculture 1959* (The United States Depa
Washington, D.C.) p. 244.] The quantity used is always 3 ou

		Food Energy (Calories)	Protein (Grams)	Fat (Grams)
BB	Beef, braised	340	20	28
HR	Hamburger	245	21	17
BR	Beef, roast	420	15	39
BS	Beef, steak	375	19	32
BC	Beef, canned	180	22	10
CB	Chicken, broiled	115	20	3
CC	Chicken, canned	170	25	7
BH	Beef heart	160	26	5
LL	Lamb leg, roast	265	20	20
LS	Lamb shoulder, roast	300	18	25
HS	Smoked ham	340	20	28
PR	Pork roast	340	19	29
PS	Pork simmered	355	19	30
BT	Beef tongue	205	18	14
VC	Veal cutlet	185	23	9
FB	Bluefish, baked	135	22	4
AR	Clams, raw	70	11	1
AC	Clams, canned	45	7	1
TC	Crabmeat, canned	90	14	2
HF	Haddock, fried	135	16	5
MB	Mackerel, broiled	200	19	13
MC	Mackerel, canned	155	16	9
PF	Perch, fried	195	16	11
SC	Salmon, canned	120	17	5
DC	Sardines, canned	180	22	9
UC	Tuna, canned	170	25	7
RC	Shrimp, canned	110	23	1

- The data used by the author are provided below.
- The principal nutrients in meat, fish, and fowl are listed.
- Recall that 1oz= 28.34952g.
- Estimated daily dietary allowances are: food energy (3200 cal), protein (70 g), calcium (0.8 g), and iron (10 mg).
- Table 4.2 convents the variables (with the exception of Fat) in percentage of food delivery.

Table 4.2 Nutrients in Meat, Fish, and Fowl

As a percentage of recommended daily allowances.

	Food Energy	Protein	Fat (Grams)	Calcium	Iron
Beef, braised	11	29	28	1	26
Hamburger	8	30	17	1	27
Beef, roast	13	21	39	1	20
Beef, steak	12	27	32	1	26
Beef, canned	6	31	10	2	37
Chicken, broiled	4	29	3	1	14
Chicken, canned	5	36	7	2	15
Beef, heart	5	37	5	2	59
Lamb leg, roast	8	29	20	1	26
Lamb shoulder, roast	9	26	25	1	25
Ham, smoked	11	29	28	1	25
Pork roast	11	27	29	1	25
Pork simmered	11	27	30	1	25
Beef tongue	6	26	14	1	25
Veal cutlet	6	33	9	1	27
Bluefish, baked	4	31	4	3	6
Clams, raw	2	16	1	10	60
Clams, canned	1	10	1	9	54
Crabmeat, canned	3	20	2	5	8
Haddock, fried	4	23	5	2	5
Mackerel, broiled	6	27	13	1	10
Mackerel, canned	5	23	9	20	18
Perch, fried	6	23	11	2	13
Salmon, canned	4	24	5	20	7
Sardines, canned	6	31	9	46	25
Tuna, canned	5	36	7	1	12
Shrimp, canned	3	33	1	12	26

- For e.g., the first (BB) ligne is obtained in the following way:
- $340/3200 = 11\%$(Food Energy).
- $20/70 = 29\%$(Protein).
- $0.009/0.8 = 1\%$ (Calcium).
- $2.6/10 = 26\%$ (Iron).
- An argument could be made that iron is less important than calories or protein and so should be given less weight or ignored entirely.
- There are n objects and k clusters, $k \leq n$.

- Our purpose is to partition the n objects (here foods) so that objects within clusters are close and objects in different clusters are distant.
- Each cluster contains at least one object and each object belongs to only one cluster.
- There is a very large number of possible partitions.

Exercice 9

What is the number of possible partitions ?

K-Means

- The discordance between the data and a given partition is denoted by e.
- We use the technique of local optimization.
- A neighborhood of partitions is defined for each partition.
- Starting from an initial partition, search through a set of partitions at each step.
- Move from the partition to a partition in its neighborhood for which e is minimal.
- If the neighborhoods are very large, it is cheaper computationally to move to the first partition discovered in the neighborhood where e is reduced from its present value.
- A number of stopping rules are possible.
- For example, the search stops when e is not reduced by movement to the neighborhood.
- The present partition is then considered locally optimal in that it is the best partition in its neighborhood.
- Consider partitions of the five ($n = 5$) beef foods {BB, BR,BS,BC,BH} into three clusters ($k = 3$).
- Totally, there are 25 such partitions.
- A plausible neighborhood for a partition is the set of partitions obtained by transferring an object from one cluster to another.
- For the partition (BB BR) (BS) (BC BH), the neighborhood consists of the following ten partitions:

1. (BR) (BB BS) (BC BH)
2. (BR) (BS) (BB BC BH)
3. (BB) (BR BS) (BC BH)
4. (BB) (BS) (BR BC BH)
5. (BB BR BS) O (BC BH)
6. (BB BR) O (BS BC BH)
7. (BB BR BC) (BS) (BH)
8. (BB BR) (BS BC) (BH)
9. (BB BR BH) (BS) (BC)
10. (BB BR) (BS BH) (BC)

K-Means Algorithm

- Let $\mathbf{x}^j \equiv (x_1^j, \cdots, x_n^j)$ the vector of values for the object j, $j = 1, \cdots, m$.
- The variables are assumed scaled.
- The partition has k disjoint clusters clusters $C_1, \cdots, C_k$, which are the indices of the objects in the various clusters.

- Let m_l be the number of objects in cluster C_l.
- Each of the m objects lies in just one of the k clusters.
- Note that $\sum_{l=1}^{k} m_l = m$.
- The vector of means over the objects in cluster C_l is denoted by $\bar{\mathbf{x}}^l$, with

$$\bar{\mathbf{x}}^l \equiv \frac{1}{m_l} \sum_{j \in C_l} \mathbf{x}^j = (\bar{x}_1^l, \cdots, \bar{x}_n^l),\ l = 1, \cdots, k,$$

  where

$$\bar{x}_i^l \equiv \frac{\sum_{j \in C_l} x_i^j}{m_l},\ i = 1, \cdots, n;\ l = 1, \cdots, k.$$

- The distance between the object j and the cluster l is $d(\mathbf{x}^j, \bar{\mathbf{x}}^l)$, where d is taken to be the Euclidian distance

$$d(\mathbf{x}^j, \bar{\mathbf{x}}^l) = ||\mathbf{x}^j - \bar{\mathbf{x}}^l|| = \left[\sum_{i=1}^{n} (x_i^j - \bar{x}_i^l)^2 \right]^{1/2},\ j = 1, \cdots, m;\ l = 1, \cdots, k.$$

  where $||\cdot||$ is the Euclidian norm.
- The error of the partition is measured by

$$e = \sum_{l=1}^{k} \sum_{j \in C_l} d^2(\mathbf{x}^j, \bar{\mathbf{x}}^l) = \sum_{l=1}^{k} \sum_{j \in C_l} ||\mathbf{x}^j - \bar{\mathbf{x}}^l||^2.$$

- Alternatively, we have

$$e = \sum_{j=1}^{m} d^2(\mathbf{x}^j, \bar{\mathbf{x}}^{l(j)}) = \sum_{j=1}^{m} \sum_{i=1}^{n} ||\mathbf{x}^j - \bar{\mathbf{x}}^{l(j)}||^2,$$

where $l(j)$ is the index of the cluster of object j.

- The general procedure is to search for a partition with a small error e by moving cases from one cluster to another.
- The search ends when no such movement reduces e.
- STEP 1. Assume initial clusters. Compute the cluster means $\bar{\mathbf{x}}^l$ and the initial error e.
- STEP 2. For the first Object, compute for every cluster l

$$\Delta e = \frac{m_l d^2(\mathbf{x}^1, \bar{\mathbf{x}}^l)}{m_l + 1} - \frac{m_{l(1)} d^2(\mathbf{x}^1, \bar{\mathbf{x}}^{l(1)})}{m_{l(1)} - 1},\ l = 1, \cdots, k,\ l \neq l(1).$$

- It corresponds to the error variation in transferring the first object from cluster $l(1)$ to which it belongs to cluster l.
- If the minimum of this quantity over all $l \neq l(1)$ is negative, transfer the first case from cluster $l(1)$ to this minimal l.
- Adjust the cluster means of $l(1)$ and the minimal l and add the error variation (which is negative) to e.
- STEP 3. Repeat Step 2 for each object j such that $2 \leq j \leq m$.
- STEP 4. lf no movement of an object from one cluster to another occurs for any case, stop. Otherwise, return to Step 2.

Exercice 10

Prove that the error variation is indeed given by:

$$\Delta e = \frac{m_l d^2(\mathbf{x}^1, \bar{\mathbf{x}}^l)}{m_l + 1} - \frac{m_{l(1)} d^2(\mathbf{x}^1, \bar{\mathbf{x}}^{l(1)})}{m_{l(1)} - 1},\ l = 1, \cdots, k,\ l \neq l(1).$$

K-MEANS APPLIED TO FOOD NUTRIENT DATA

- Only the first eight foods will be considered.
- Only three variables, food energy, protein, and calcium as a percentage of recommended daily allowances are used.
- The eight foods ($m = 8$) are partitioned into three clusters ($k = 3$).

Table 4.3 Application of *K*-Means Algorithm to Food Data

	Energy	Protein	Calcium
BB	11	29	1
HR	8	30	1
HR	13	21	1
BS	12	27	1
BC	6	31	2
CB	4	29	1
CC	5	36	1
BH	5	37	2

INITIAL CLUSTER MEANS				e = 154.9
	Energy	Protein	Calcium	
1. HR, CB	8.5	25	1	
2. HR, BS	10	28.5	1	
3. BB, BC, CC, BH	6.75	33.25	1.5	

FIRST CHANGE				e = 108.2
1. HR, CB	8.5	25	1	
2. HR, BS, BB	10.33	28.67	1	
3. BC, CC, BH	5.33	34.67	1.67	

SECOND CHANGE				e = 61.4
1. HR	13	21	1	
2. HR, BS, BB	10.33	28.67	1	
3. BC, CC, BH, CB	5	33.25	1.5	

«««< HEAD

```
#library("cluster.datasets")
#write.csv(rda.meat.fish.fowl.1959,"Hartigandat%a1.csv")
df<-read.csv("Hartigandata1.csv")
print(df)
```

```
##     X               name energy protein fat calcium iron
## 1   1       Braised beef     11      29  28       1   26
## 2   2          Hamburger      8      30  17       1   27
```

```
## 3   3          Roast beef     18      21  39       1   20
## 4   4           Beefsteak     12      27  32       1   26
## 5   5         Canned beef      6      31  10       2   37
## 6   6     Broiled chicken      8      29   3       1   14
## 7   7      Canned chicken      5      36   7       2   15
## 8   8          Beef heart      5      37   5       2   59
## 9   9      Roast lamb leg      8      29  20       1   26
## 10 10 Roast lamb shoulder      9      26  25       1   23
## 11 11          Smoked ham     11      29  28       1   25
## 12 12          Pork roast     11      27  29       1   25
## 13 13       Pork simmered     11      27  30       1   24
## 14 14         Beef tongue      6      26  14       1   25
## 15 15         Veal cutlet      6      33   9       1   27
## 16 16      Baked bluefish      4      31   4       3    6
## 17 17           Raw clams      2      16   1      10   60
## 18 18        Canned clams      1      10   1       9   54
## 19 19     Canned crabmeat      3      20   2       5    8
## 20 20       Fried haddock      4      23   5       2    5
## 21 21    Broiled mackerel      6      27  13       1   10
## 22 22     Canned mackerel      5      23   9      20   18
## 23 23         Fried perch      6      23  11       2   13
## 24 24       Canned salmon      4      24   5      20    7
## 25 25     Canned sardines      6      31   9      46   25
## 26 26         Canned tuna      5      36   7       1   12
## 27 27       Canned shrimp      3      33   1      12   26
```

```
df<-df[1:8,c(3,4,6)]
df
```

```
##   energy protein calcium
## 1     11      29       1
## 2      8      30       1
## 3     18      21       1
## 4     12      27       1
## 5      6      31       2
## 6      8      29       1
## 7      5      36       2
## 8      5      37       2
```

```
# The data contain some errors
df[3,1]<-13 # Error in ligne 3
df[6,1]<-4 # Error in ligne 6
df[7,3]<-1 # Error at ligne 7
df
```

```
##   energy protein calcium
## 1     11      29       1
## 2      8      30       1
## 3     13      21       1
## 4     12      27       1
## 5      6      31       2
## 6      4      29       1
## 7      5      36       1
## 8      5      37       2
```

```
rownames(df)<-c("BB","HR","BR","BS","BC","CB","CC","BH")
df
```

```
##    energy protein calcium
## BB     11      29       1
## HR      8      30       1
## BR     13      21       1
## BS     12      27       1
## BC      6      31       2
## CB      4      29       1
## CC      5      36       1
## BH      5      37       2
```

```
colnames(df)<-c("Energy","Protein","Calcium")
df
```

```
##    Energy Protein Calcium
## BB     11      29       1
## HR      8      30       1
## BR     13      21       1
## BS     12      27       1
## BC      6      31       2
## CB      4      29       1
## CC      5      36       1
## BH      5      37       2
```

- STEP 1. An initial clustering which works well is based on the object sums.
- We denote by s_j the onject sum of object j, $j = 1, \cdots, m$.
- We set the object j to the cluser $l(j)$ such that

$$l(j) = \left[k \left(\frac{s_j - \min_{l=1\cdots k} s_l}{\max_{l=1\cdots k} s_l - \min_{l=1\cdots k} s_l} \right) + 1 \right],$$

where $[\cdot]$ is the integer part function.

```
s=rowSums(df)
cluster=trunc(3*(s-min(s))/(max(s)-min(s))+1)
cluster
```

```
## BB HR BR BS BC CB CC BH
##  3  2  1  2  2  1  3  4
```

```
Myfunc<-function(x){m=min(x,3)
return(m)}
cluster<-sapply(cluster, Myfunc)
cluster
```

```
## BB HR BR BS BC CB CC BH
##  3  2  1  2  2  1  3  3
```

- Note that objets whith minimal sum are attributed to cluster $k+1$ with this rule and must be attributed to k.

- Thus the initial partition shoul be: (BR CB) (HR BS BC) (BB CC BH).

```
rownames(df)[cluster==1]
```

```
## [1] "BR" "CB"
```

```
rownames(df)[cluster==2]
```

```
## [1] "HR" "BS" "BC"
```

```
rownames(df)[cluster==3]
```

```
## [1] "BB" "CC" "BH"
```

- Hartigan has some errors since he found sums equal to $41, 39, 35, 40, 41, 34, 42, 44$.

```
s=c(41,39,35,40,41,34,42,44)
3*(s-min(s))/(max(s)-min(s))+1
```

```
## [1] 3.1 2.5 1.3 2.8 3.1 1.0 3.4 4.0
```

```
cluster<-trunc(3*(s-min(s))/(max(s)-min(s))+1)
cluster
```

```
## [1] 3 2 1 2 3 1 3 4
```

```
cluster<-sapply(cluster,Myfunc)
cluster
```

```
## [1] 3 2 1 2 3 1 3 3
```

- The corresponding clusters are 3, 2, 1, 2, 3, 1, 3, and 3.
- Thus the initial partition is (BR CB) (HR BS) (BB BC CC BH).

```
rownames(df)[cluster==1]
```

```
## [1] "BR" "CB"
```

```
rownames(df)[cluster==2]
```

```
## [1] "HR" "BS"
```

```
rownames(df)[cluster==3]
```

```
## [1] "BB" "BC" "CC" "BH"
```

- We then compute the means within each cluster.

```
colMeans(df[cluster==1,])
```

```
##  Energy Protein Calcium
##     8.5    25.0     1.0
```

```
colMeans(df[cluster==2,])
```

```
##  Energy Protein Calcium
##    10.0    28.5     1.0
```

```
colMeans(df[cluster==3,])
```

```
##  Energy Protein Calcium
##    6.75   33.25    1.50
```

- For example, the mean of objects in the first cluster for the first variable, equals $(13 + 4)/2 = 8.5$ (See Table 4.3 for more.)
- The error for the initial partition is the sum of squared distances of cases from their cluster means.

```
MatrixC1=rbind(df[cluster==1,],colMeans(df[cluster==1,]))
#rownames(MatrixC1)[3]<-c("MeanC1")
#MatrixC1
#sum((as.matrix(D)[3,])^2)
```

```
#MatrixC2=rbind(df[cluster==2,],colMeans(df[cluster==2,]))
#MatrixC2
#rownames(MatrixC2)[3]<-c("MeanC2")
#MatrixC3=rbind(df[cluster==3,],colMeans(df[cluster==3,]))
#rownames(MatrixC3)[5]<-c("MeanC3")
#MatrixC3
```