Distance and dissimilarities

Table of Contents

knitr::opts\_chunk$set(echo = TRUE)  
#install.packages("dplyr")  
#install.packages("stargazer")  
#install.packages("ade4")  
#install.packages("magrittr")  
#install.packages("cluster")

# Definition of a distance

* A distance function or a metric on , is a function .
* A distance function must satisfy some required properties or axioms.
* There are three main axioms.
* A1. (identity of indiscernibles);
* A2. (symmetry);
* A3. (triangle inequality), where , and are all vectors of .
* 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:

# Euclidean distance

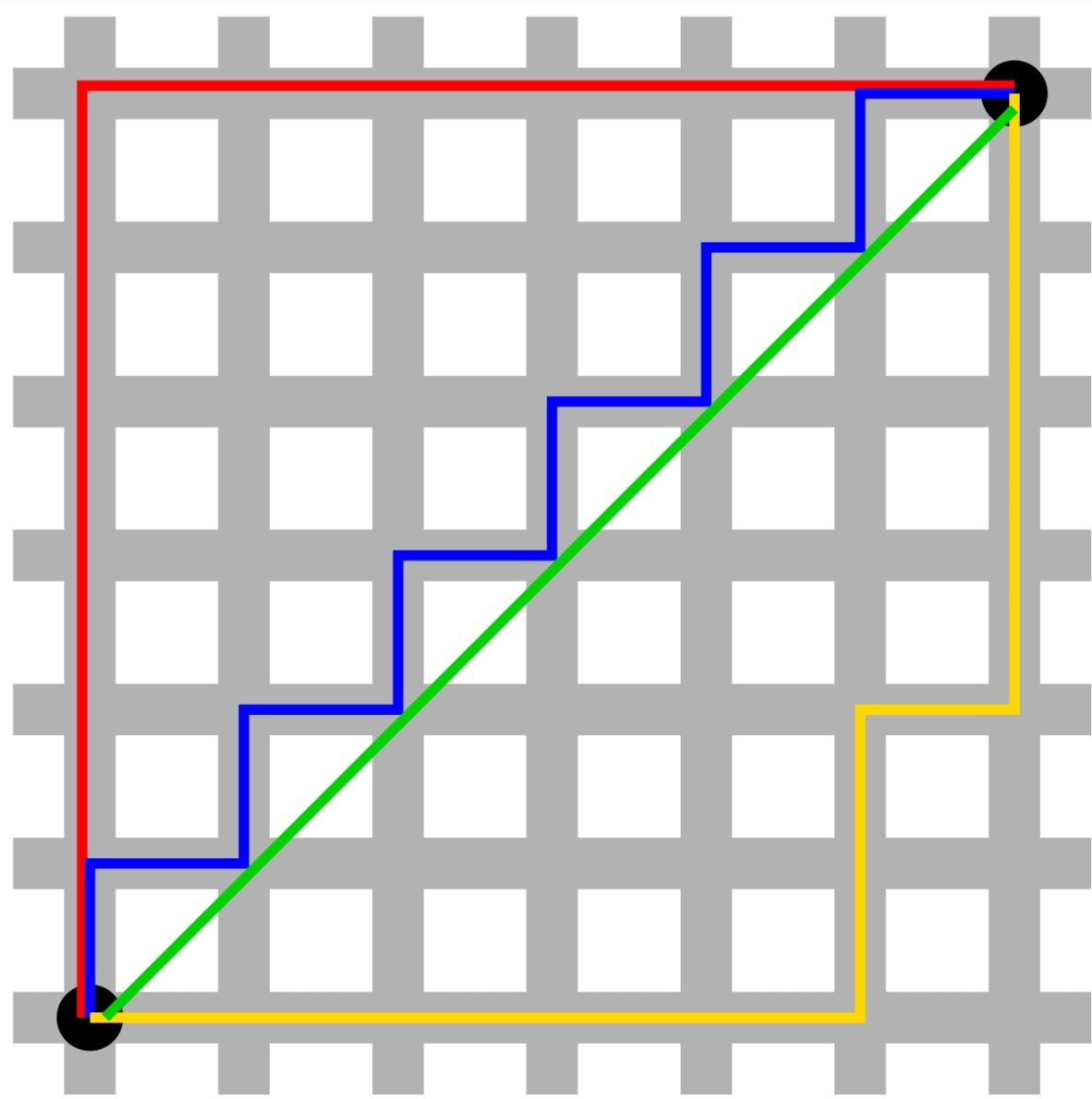
* It is defined by:
* A1-A2 are obvious.
* The proof of A3 is provided below.

# 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:
* A1-A2 hold.
* A3 also holds using the fact that for any reals .
* 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

6\*sqrt(2)

## [1] 8.485281

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

## x  
## y 12

# Canberra distance

* It is defined by:
* Note that the term is not properly defined as:when .
* By convention we set the ratio 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 32

* Prove that the Canberra distance is a true distance.

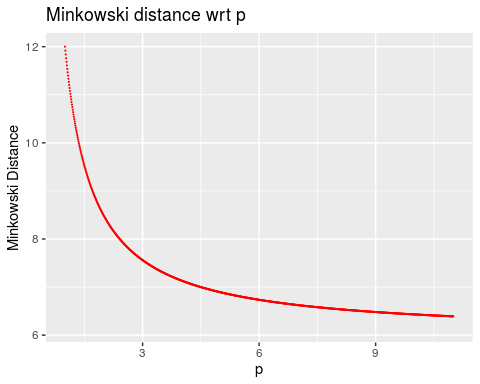
# Minkowski distance

* Both the Euclidian and the Manattan distances are special cases of the Minkowski distance which is defined, for , by:
* For , we get the Manhattan distance.
* For , we get the Euclidian distance.
* Let us also define:
* where is known as the -norm or Minkowski norm.
* Note that the Minkowski distance and norm are related by:
* Conversely, we have:

where is the null-vetor of .

library("ggplot2")  
x = c(0, 0)  
y = c(6,6)  
MinkowDist=c()  
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")+xlim(1,11)+xlab("p")+ylab("Minkowski Distance")+ggtitle("Minkowski distance wrt p")

## Warning: Removed 1900 rows containing missing values (geom\_point).



# Chebyshev distance

* At the limit, we get the Chebyshev distance which is defined by:
* The corresponding norm is:

# Minkowski inequality

* The proof of the triangular inequality A3 is based on the Minkowski inequality:
* For any nonnegative real numbers ; , and for any , we have:
* To prove that the Minkowski distance satisfies A3, notice that
* Since for any reals , we have: , and using the fact that is increasing in , we obtain:
* Applying the Minkowski inequality with and , , we get:

# Hölder inequality

* The proof of the Minkowski inequality itself requires the Hölder inequality:
* For any nonnegative real numbers ; , and any with , we have:
* The proof of the Hölder inequality relies on the Young inequality:
* For any , we have
* with equality occuring iff: .
* To prove the Young inequality, one can use the (strict) convexity of the exponential function.
* For any reals , we have:
* We then set: and 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:
* Note that for , we have . The Hölder inequality implies for that special case
* Since the LHS od thes above inequality is greater then , we get the Cauchy-Schwartz inequality

\* Using the dot product notation called also scalar product noation: , and the norm notation , the Cauchy-Schwart inequality is:

# Pearson correlation distance

* The Pearson correlation coefficient is a similarity measure on defined by:
* where is the mean of the vector defined by:
* Note that the Pearson correlation coefficient satisfies P2 and is invariant to any positive linear transformation, i.e.:
* for any .
* The Pearson distance (or correlation distance) is defined by:
* Note that the Pearson distance does not satisfy A1 since for any non-zero vector . It neither satisfies the triangle inequality. However, the symmetry property is fullfilled.

# Cosine correlation distance

* The cosine of the angle between two vectors and is a measure of similarity given by:
* Note that the cosine of the angle between the two centred vectors and coincides with the Pearson correlation coefficient of and , where is a vector of units of .
* The cosine correlation distance is defined by:
* 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:
* Here, is the rank of among the set of values of .
* We illustrate this transformation with a simple example:
* If , then the rank-order vector is .

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

## [1] 2 1 3 4 5

* The Spearman’s rank correlation of two numerical variables and is simply the Pearson correlation of the two correspnding rank-order variables and , i.e. . This measure is is useful because it is more robust against outliers than the Pearson correlation.
* If all the ranks are distinct, it can be computed using the following formula:
* where .
* The spearman distance is then defined by:
* It can be shown that easaly that it is not a proper distance.
* If all the ranks are distinct, we get:

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

# Kendall tau distance

* The Kendall rank correlation coefficient is calculated from the number of correspondances between the rankings of and the rankings of .
* The number of pairs of observations among observations or values is:
* The pairs of observations and are said to be *concordant* if:
* and to be *discordant* if:
* where returns for positive numbers and negative numbers and otherwise.
* If or (or both), there is a tie.
* The Kendall coefficient is defined by (neglecting ties):
* Let (resp. ) be the number of concordant (resp. discordant) pairs, we have
* The Kendall tau distance is then:
* 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

# 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:
* where is the sample standard deviation given by:
* The transformed variable will have a mean of and a variance of .
* 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

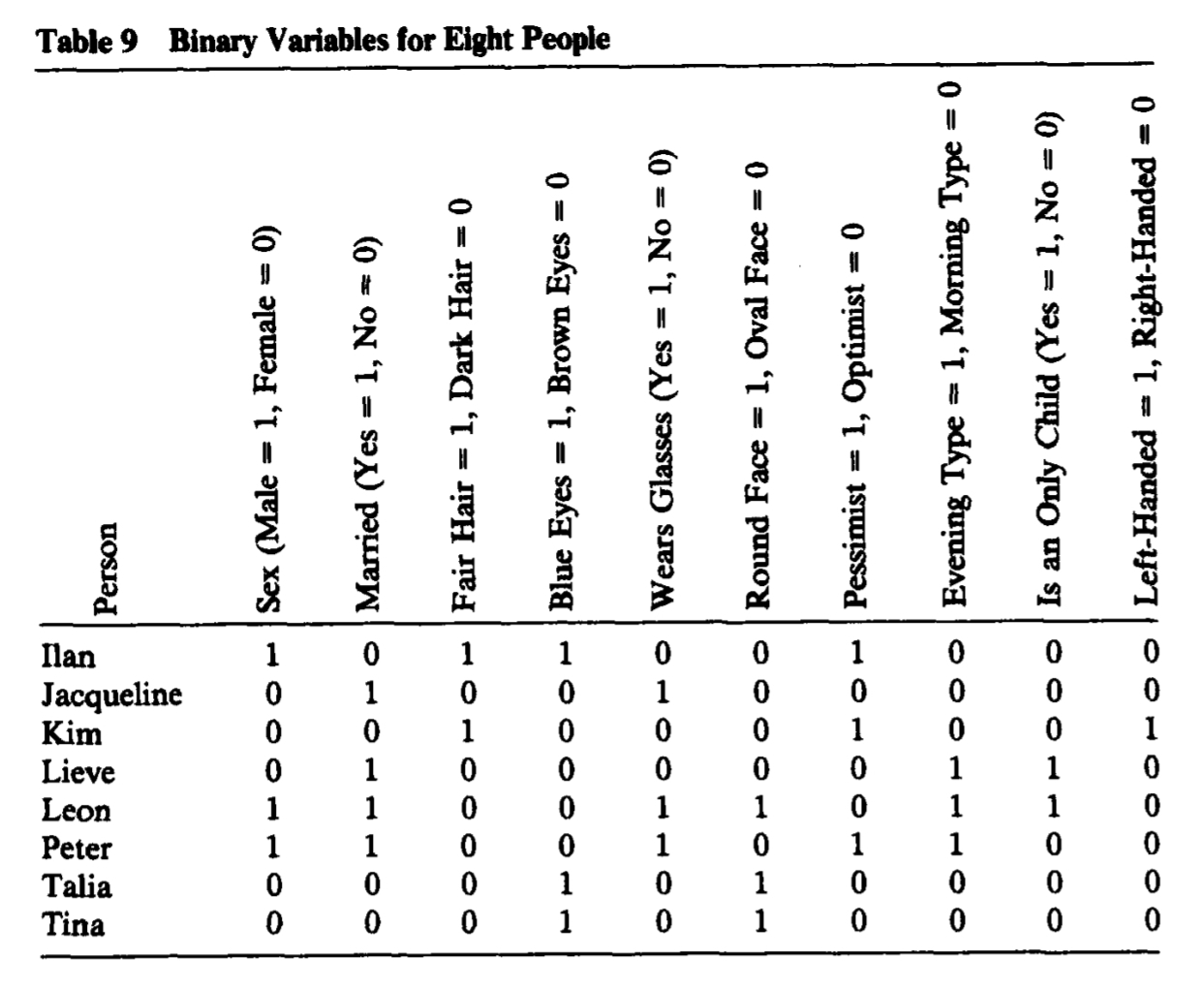
scale(y)

## [,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 characters.
* \*The presence of the character is coded by and the absence by 0.
* We have have at our disposal two vectors.
* is observed for a first individual (or object).
* is observed for a second individual.
* We can then calculate the following four statistics:
* The counts of matches are for and for ;
* The counts of mismatches are for and for .
* Note that obviously: .
* This gives a very useful association table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Second individual |  |  |
|  |  | 1 | 0 | *Totals* |
| **First individual** | 1 |  |  |  |
|  | 0 |  |  |  |
| *Totals* |  |  |  |  |

 *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 people (individuals) and 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", "Evening Type", "Is an Only Child", "Left-Handed")

* 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: .
* Note that interchanging Ilan and Talia would permute and while leaving and unchanged.
* A good similarity or dissimilarity coefficient must treat and symmetrically.
* A similarity measure is denoted by: .
* The corresponding distance is then defined as:
* Alternatively, we have:
* A list of some of the similarity measures 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 |  |  |
| Simple matching |  |  |
| Jaccard |  |  |
| Rogers and Tanimoto (1960) |  |  |
| Gower and Legendre (1986) |  |  |
| Gower and Legendre (1986) |  |  |

* To calculate these coefficients, we use the function: [dist.binary().](https://www.rdocumentation.org/packages/ade4/versions/1.7-16/topics/dist.binary)
* All the distances in this package are of type .

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)

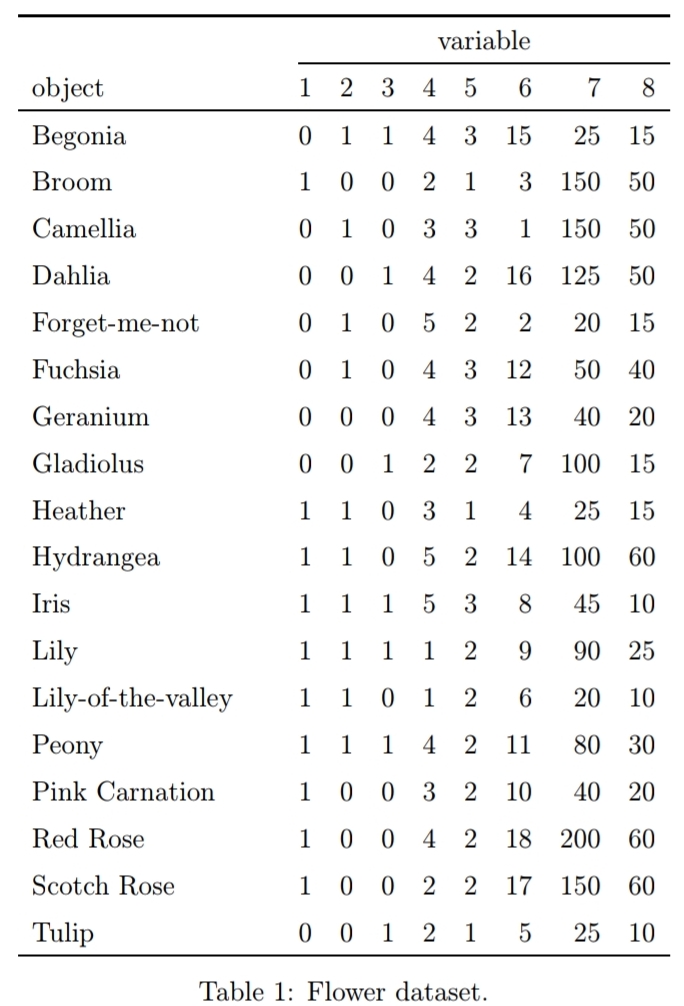
[1] 0.6666667 \* The reason for such a large number of possible measures has to do with the apparent uncertainty as to how to deal with the count of zero-zero matches . \* The measues embedding are sometimes called symmetrical. \* The other measues are called assymmetrical. \* In some cases, of course, 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 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 4 \* Prove that the distances based on the SimplemMatching 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.](https://drive.google.com/file/d/1PUQ7g9HIwwUG0CXbCsLA03hnApWMhjka/view?usp=drivesdk)

# Nominal variables

* We previuosly studied above binary variables which can only take on two states coded as .
* 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 be the number of states and code the outcomes as .
* We could choose and .
* These states are not ordered in any way
* One strategy would be creating a new binary variable for each of the nominal states.
* Then to put it equal to if the corresponding state occurs and to 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 are both nominal records for two individuals,
* Let define the function:
* Let be the number of attributes of the two individuals on which the two records match:
* Let be the number of attributes on which the two records do not match:
* Let be the number of attributes on which the two records match in a “not applicable” category:
* The distance corresponding to the simple matching approach is:
* Therefore:
* 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:
* The wheight is put equal to when both measurements and are nonmissing,
* The number is the contribution of the th measure or variable to the dissimilarity measure.
* It the th measure is nominal, we take
* If the th measure is interval-scaled, we take instead:
* where is the range of variable over the available data.
* Consider the following data set:

 *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,]

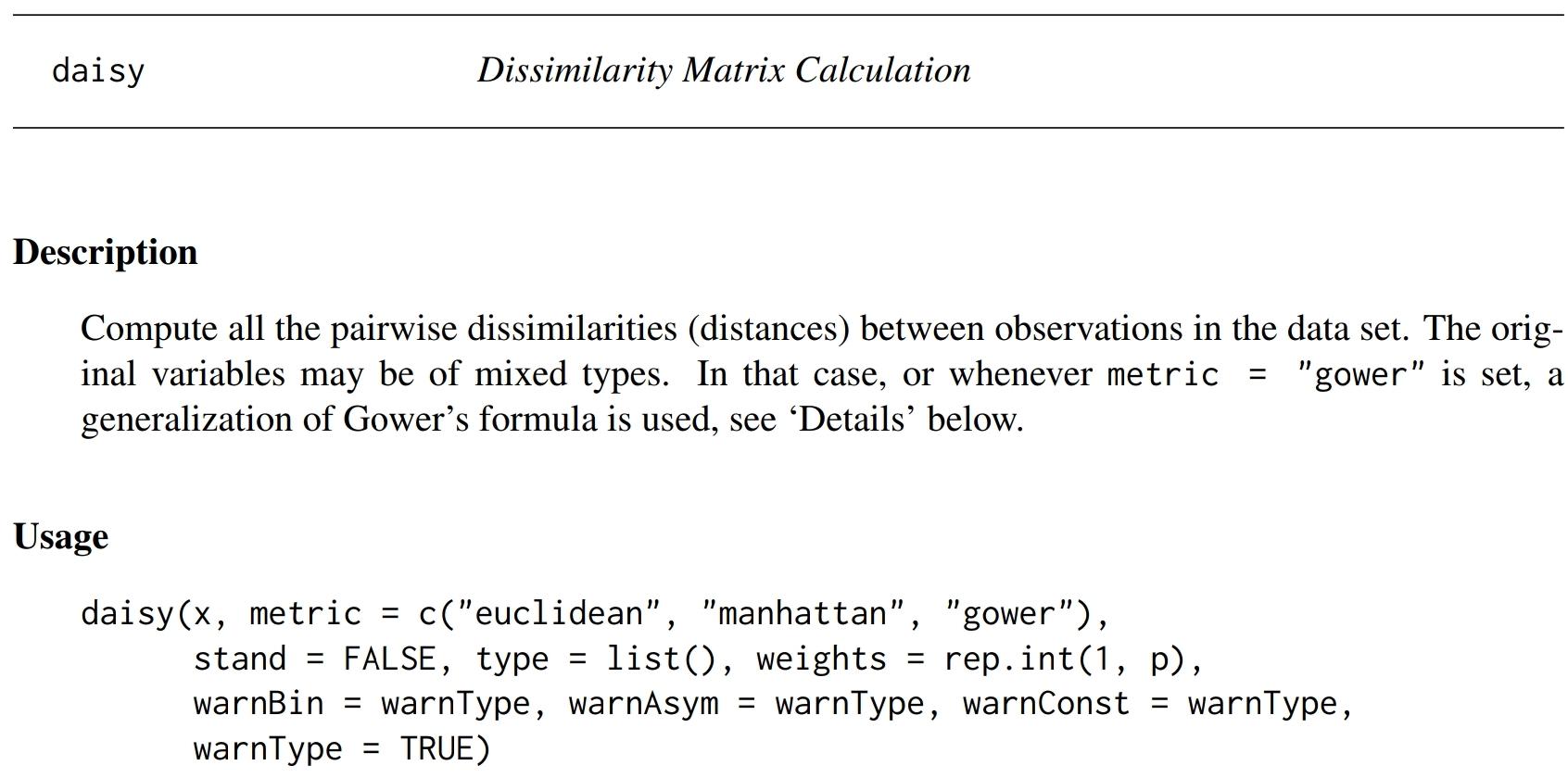
## V1 V2 V3 V4 V5 V6 V7 V8  
## 1 0 1 1 4 3 15 25 15  
## 2 1 0 0 2 1 3 150 50

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

## [1] 180

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

## [1] 50



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

KKK –>