

DM583

# Data Mining

## Data Representation

# PART 1

---

Preliminaries

# What is Data in the First Place?

- Broadly speaking, **Data** is any collection of numeric or symbolic values
- Data can be associated with quantitative or qualitative **variables** that describe quantities or properties of the world and whose values can vary
  - The temperature across different locations
    - Or the temperature at a certain location over time
  - The skin colour across different patients
  - The insured car make or price across different customers
- Data is said to be **structured** when it is organized in a pre- and well-defined form
  - otherwise, it is said to be **unstructured**

# Structured and Unstructured Data

- Data sets in many sectors of industry, government, and society are collected and stored in a **structured** way
  - For instance, clients of a bank have their personal as well as financial information (name, data of birth, income, debt, etc.) mostly collected and stored in structured databases
  - This can make data management and information retrieval easier
    - For example, traditional relational database modelling and SQL querying
- Historically, structured data largely prevailed for decades
- Since the 1990s, however, with the advent of the internet and the digital era, **unstructured** data has become more and more common
  - For instance, unprocessed text documents, webpages

# Structured and Unstructured Data

- Despite the emergence of unstructured data in various fields, when it comes to data analysis the study of structured data remains fundamentally important
  - Not only because structured databases commonly appear in many real-world problems, but because, as we shall see, many forms of unstructured data can often be pre-processed into structured data for simpler analysis (and management)
    - This way, analysis of unstructured data can often be performed in a two-stage approach
      - Conversion from unstructured to structured data (1<sup>st</sup> stage) + analysis of structured data (2<sup>nd</sup> stage)
    - The first stage can be highly domain-specific
      - It may require domain-specific expertise, e.g. signal (sound, image) or natural language processing techniques
      - This highlights the importance of interdisciplinary communication/collaboration skills for Data Scientists
  - A very common way to represent structured data is in tabular form, discussed next

# PART 2

---

Tabular Data

```
> ?CO2           # Check dataset CO2 from Base R
> data("CO2")    # Load dataset CO2
> CO2[40:45,]    # Show rows 40 to 45 (6 records)
```

# Tabular Data

- The most usual form of structured data is called **tabular** or **record** data
- In this representation, the data is organized in a matrix or table
  - possibly multiple tables, such as in relational databases
- Example:

Rows 40 to 45 of dataset "CO2" from Base R (default package "datasets")		Plant	Type	Treatment	Concentration	Uptake
	40	Qc3	Quebec	chilled	500	38.9
	41	Qc3	Quebec	chilled	675	39.6
	42	Qc3	Quebec	chilled	1000	41.4
	43	Mn1	Mississippi	nonchilled	95	10.6
	44	Mn1	Mississippi	nonchilled	175	19.2
	45	Mn1	Mississippi	nonchilled	250	26.2

# Tabular Data

- Each **row** represents a **record, observation, example, instance, or case**
  - Depending on the domain (e.g. in *data mining*), it can also be called a **data object** or **data point**
  - It can be e.g. a patient, client, customer, a credit card transaction, etc.
- Each **column** represents a **variable** or **attribute**
  - Properties or characteristics that describe the records
  - For example, age, income, symptoms, value of transaction, date of transaction, etc.

		Plant	Type	Treatment	Concentration	Uptake
Rows (records/observations)	40	Qc3	Quebec	chilled	500	38.9
	41	Qc3	Quebec	chilled	675	39.6
	42	Qc3	Quebec	chilled	1000	41.4
	43	Mn1	Mississippi	nonchilled	95	10.6
	44	Mn1	Mississippi	nonchilled	175	19.2
	45	Mn1	Mississippi	nonchilled	250	26.2
		Columns (variables/attributes)				



# Types of Variables

- Variables can be Categorical or Numerical
- **Categorical** variables are qualitative descriptions
  - They can be Nominal or Ordinal
  - **Nominal Variables**
    - Merely qualitative descriptions that carry no order on their values
    - For example, color (green, blue, yellow, ...), marital state (single, married, divorced, widowed, ...)
      - The variable is called **binary** if it takes only two possible values: e.g., sex\* (male, female)
    - In the CO2 dataset, variables Type and Treatment are nominal variables

# Types of Variables

- Variables can be Categorical or Numerical
- **Categorical** variables are qualitative descriptions
  - They can be Nominal or Ordinal
  - **Ordinal Variables**
    - Qualitative descriptions whose values follow an order
    - Examples:
      - Degree of education: Primary school < Secondary school < High school < Bachelor's < Master's < PhD
      - Month of the year: January < February < ... < December
      - Levels of Pain: No < Mild < Moderate < Severe
      - Qualitative Height: Very Short < Short < Average < Tall < Very Tall

# Types of Variables

- Variables can be Categorical or Numerical
- **Numerical** variables are quantitative descriptions
  - They can be Discrete or Continuous
  - **Discrete variables**
    - They can only assume an enumerable number of values, e.g., integers (... , -2, -1, 0, 1, 2, ...)
    - Often, they appear as **counting variables**
      - Non-negative integers, for instance, number of children (0, 1, 2, 3), number of words in a text, etc.
      - A particular case is the **binary** numerical variable, which takes only two values
        - For instance, 0 and 1, to numerically represent the absence (0) or presence (1) of a certain property

# Types of Variables

- Variables can be Categorical or Numerical
- **Numerical** variables are quantitative descriptions
  - They can be Discrete or Continuous
  - **Continuous variables**
    - They can conceptually take on any value from the set of **real numbers** or a subset/interval of it
    - Examples:
      - Temperature, Height, Elapsed Time, etc.
    - In the CO<sub>2</sub> example, variables Concentration and Uptake are continuous

# Types of Variables

- Operations that are meaningful for different types
  - **Nominal variables:** only accept equality/inequality comparisons
    - $=$  and  $\neq$  ( $A = B?$   $A \neq B?$ )
  - **Ordinal variables:** in addition, also accept order comparisons
    - $=, \neq, <, >, \leq, \geq$
  - **Numerical variables:** in addition, also accept arithmetic operations
    - $=, \neq, <, >, \leq, \geq, +, -, *, /$
    - Note: certain restrictions may apply to discrete numerical values.
      - For instance, integer variables should only cope with integer divisions (e.g.,  $10/3$  shall yield 3)

# Types of Variables

- Hypothetical example of tabular data:

Name	Temp.	Nausea	Rash Area	Pain	Exam X	Result
Chloe	38.3	yes	large	no	300	Negative
Mary	37.8	no	large	yes	450	Positive
Anne	37.4	no	large	yes	420	Negative
Leah	37.0	no	small	no	550	Negative
James	37.7	yes	small	yes	500	Positive
John	39.1	no	medium	yes	1000	Positive

- **Exercise:** Identify the types of variables in the above example assuming that Exam X is measuring the amount of certain cells per unit of volume and cannot be fractional

# Types of Variables

- **Exercise:** Identify appropriate types for the following variables
  - Number of certain blood cells in a blood sample
  - The relative rank of each student of a class with respect to their final grades
  - Length of an object as measured to any desired precision
  - The hair color of an individual
  - The postal code of an individual
- **Hint:** think not only of the values that they possibly take, but also the operations that make sense for each of them

# Types of Variables

- An important distinction that we shall discuss later is between variables/attributes that are symmetric as opposed to those that are asymmetric
  - A variable is **asymmetric** if only certain values are informative
    - Typically, a numerical asymmetric variable is one for which zeros are not relevant
      - This relates to the concept of **sparse datasets** (datasets for which most data entries are null)
    - Only non-zero values are relevant as they represent the presence of something
    - For instance, a word or item that may (1) or not (0) appear in documents or sales transactions
      - Documents / transactions are characterized by the words / items that they contain (1's)
        - They are not characterized by things that they do not contain, so zeros do not matter
  - A variable is **symmetric** if values are equally important
    - The interpretation is context-dependent, but apart from context, sex is usually a good example of symmetric variable



# Types of Variables

- In R, the type **factor** represents categorical variables
  - Factors take on a limited, pre-defined set of different values, which may or not be ordered
- **Nominal Variables**
  - Nominal variables are defined as **non-ordered** factors, such as in the example below:

```
> mycolors = c("blue","yellow","red")
> fac <- factor(levels = mycolors, ordered = FALSE) # Declare a non-ordered factor that can take on three values (colours)
> fac[1:4] = c("yellow","red","blue","red") # Assign 4 different values to the variable as a vector
> fac[1] == fac[2] # FALSE
> fac[2] == fac[4] # TRUE
> try(fac[1] > fac[2]) # Warning message
> is.factor(fac) # TRUE
> is.ordered(fac) # FALSE
```

- If the variable is binary, it can also be defined as of type **logical** in R (FALSE or TRUE)

# Types of Variables

```
> data(CO2)
> is.ordered(CO2$Plant)
TRUE
> levels(CO2$Plant)
[1] "Qn1" "Qn2" "Qn3" "Qc1" "Qc3" "Qc2"
"Mn3" "Mn2" "Mn1" "Mc2" "Mc3" "Mc1"
```

- In R, the type **factor** represents categorical variables
  - Factors take on a limited, pre-defined set of different values, which may or not be ordered
- **Ordinal Variables**
  - Ordinal variables can be defined as **ordered** factors, such as in the example below:

```
> months = c("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December")
> fac <- factor(c("March", "March", "January", "July"), levels = months, ordered = TRUE)
> fac[1] > fac[2] # FALSE
> fac[1] > fac[3] # TRUE
> fac[1] > fac[4] # FALSE
> is.factor(fac) # TRUE
> is.ordered(fac) # TRUE
```

- The variable Plant in the CO2 dataset is defined as ordinal (see top-right corner)

# Types of Variables

- **Numeric variables** in R can be easily represented using basic types
  - The “**numeric**” type can be used to store both continuous as well as discrete variables
  - The “**integer**” type can also be used to more efficiently store discrete (e.g. counting) variables
- Examples:

```
x <- 10                # x = 10.00
is.integer(x)          # FALSE
is.numeric(x)          # TRUE
x <- x/3               # x = 3.333...
x <- 10L               # x = 10 (L is used to store as integer)
is.integer(x)          # TRUE
x <- 10L/%3            # x = 3.0 (integer division but result returned as numeric)
is.integer(x)          # FALSE
x <- as.integer(10/3)  # x = 3 (same outcome is obtained from x <- 10L/%3L)
is.integer(x)          # TRUE
```

# Tabular Data

- Data tables with different types of variables
  - Represented in R via a **Data Frame**
  - Example:

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tan, Steinbach, Kumar, "Introduction to Data Mining", Addison Wesley, 2006 [1]

```
> Refund <- c(TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE)
> MStatus <- factor(levels = c("Single","Married","Divorced"), ordered = FALSE)
> MStatus[1:10] <- c("Single", "Married", "Single", "Married", "Divorced", "Married", "Divorced", "Single", "Married", "Single")
> TIncome <- c(125000, 100000, 70000, 120000, 95000, 60000, 220000, 85000, 75000, 90000)
> Cheat <- c(FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE)
> DfTanEtAl <- data.frame(Refund = Refund, Marital_Status = MStatus, Taxable_Income = TIncome, Cheat = Cheat)
```

- Alternatively, a **matrix** can also be used when all variables are of the same type
  - In particular, for matrices with numerical variables only, mathematical operators are applicable

# Tabular Data

- Same example as before, but now reading from a CSV file and making adjustments:

```
Refund,MStatus,TIncome,Cheat
Y,Single,125000.0,N
N,Married,100000.0,N
N,Single,70000.0,N
Y,Married,120000.0,N
N,Divorced,95000.0,Y
N,Married,60000.0,N
Y,Divorced,220000.0,N
N,Single,85000.0,Y
N,Married,75000.0,N
N,Single,90000.0,Y
```

Example\_Tan\_Etal.csv

```
> test <- read.csv("./R/MyFiles/Example_Tan_Etal.csv") # Read CSV file with comma separated values and period as decimal separator
> names(test) <- c("Refund", "Marital_Status", "Taxable_Income", "Cheat") # Change variable names
> test$Refund <- factor(test$Refund, levels = c("Y","N"), labels = c("Yes", "No")) # Change categorical values for Refund
> test$Cheat <- factor(test$Cheat, levels = c("Y","N"), labels = c("Yes", "No")) # Change categorical values for Cheat
> test
```

	Refund	Marital_Status	Taxable_Income	Cheat
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes
6	No	Married	60000	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes
9	No	Married	75000	No
10	No	Single	90000	Yes

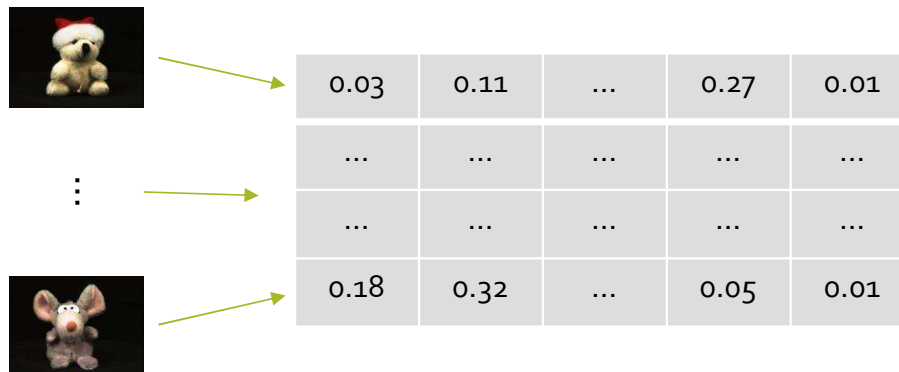
# Tabular Data: Matrices

- As we mentioned before, different types of data can be structured in a tabular form. In the following we discuss some examples involving numerical matrices
- **Images:**
  - An image can be represented as a matrix of pixels, which in turn can be described by a number that quantifies the colour of that pixel within a colour spectrum
    - Therefore, an image can be represented in a tabular form, as a numerical (real-valued) matrix
  - However, this raw representation is not necessarily the best one for every type of analysis
  - In the field of image processing, there are several feature extraction techniques that allow to extract a collection of features that describe relevant characteristics of an image
    - These features are a “signature” of the image, e.g., in terms of its colours, shapes, texture, etc.

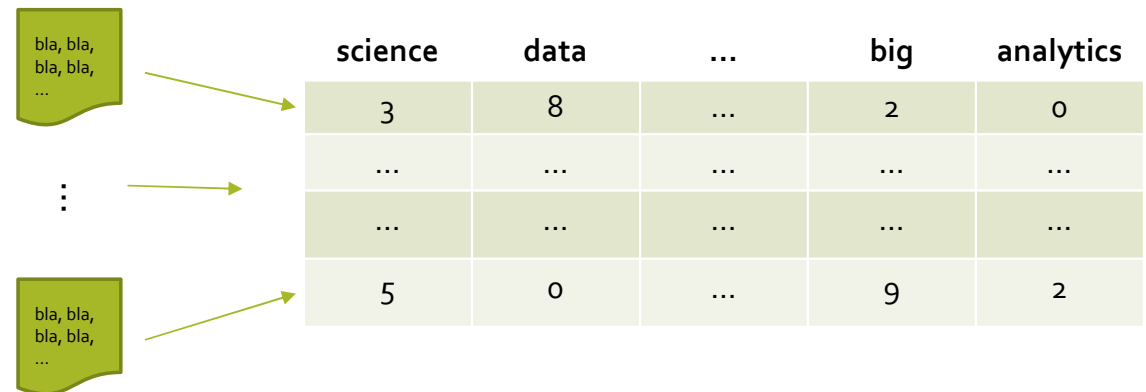
# Tabular Data: Matrices

- **Images – continued:**

- By using one (or a combination) of these feature extraction techniques, an image can be described as a numerical vector, i.e., a one-dimensional real-valued array
- A collection of images can, therefore, be represented as a matrix where each row describes an image, and each column describes a feature (variable/attribute) of the image



# Tabular Data: Matrices



- **Text:**

- After some pre-processing, which includes the removal of irrelevant terms (such as articles, prepositions, etc.), a text document can be represented as a numerical vector
  - In one of the most common representations, called **bag-of-words**, this vector is called **term-vector**
  - Its elements are the counts or relative frequencies of relevant words in the document
- A collection of documents can therefore be represented as a numerical matrix where each row describes a document, and each column describes a term of the documents (e.g. **TF-IDF**)
  - Notice that “document” can be interpreted broadly. For instance, it can refer to the textual portion of **webpages**, **tweets** in twitter, **e-mails**, on-line product **reviews**, etc.



# Note on Matrices and Embeddings

- Complex data objects such as images and unstructured text can also be described in suitable vectorized/matrix form using modern machine learning representations called **embeddings**
- Roughly speaking, an embedding is a vectorized numerical representation of an object such as an image, video-frame or text, which is learnt from examples (training data) to maximize performance in a given machine learning (ML) context
- Learning usually takes place in *supervised* or *self-supervised* fashion
- An embedding can be, e.g., a representation layer of a *deep neural network*
  - CNNs, RNNs, Auto-encoders

# Note on Matrices and Embeddings

- Embeddings can be specifically **trained** using the target dataset of interest, or they can also be used *off-the-shelf* (reusing models **pre-trained** for similar or related tasks)
  - For instance, one can remove the final SoftMax layer of a deep CNN trained for *image* classification and take the remaining deepest layer as an image embedding
  - For *text*, there are specific embedding models to represent "documents", e.g., `doc2vec`
    - See, for example: [Doc2vec package documentation in R \(CRAN Repository\)](#)
  - There are also embedding models to represent individual words, e.g., `word2vec`
    - See, for example: [Word2vec package documentation in R \(CRAN Repository\)](#)
    - These will be particularly relevant when discussing numeric conversion of categorical variables

# Tabular Data: Matrices

User/Item	Item 1	Item 2	...	Item n
User 1	5	0	...	3
User 2	0	4	...	0
...	...	...	...	...
User N	0	0	...	2

- **Transactions:**

- Certain transactional data can be conceptually represented as matrices
- For instance, online customers buying or rating products
- Each row describes a user, and each column describes an item (product)
  - **User-Item matrix** entries can be binary or non-binary, e.g.:
    - Binary: user has or hasn't purchased / "liked" an item (1 or 0), or
    - Non-binary: user has or hasn't rated a product (1 – 5 score or 0)
  - This is a common representation in *recommender systems*
  - **Note:** since these matrices are usually large and sparse, they are normally internally represented using especial data structures for sparse matrices

# Tabular Data: Matrices

	Time 1	Time 2	...	Time n
Gene 1	0.34	0.01	...	2.67
Gene 2	0.05	4.08	...	1.36
...	...	...	...	...
Gene N	0.00	0.01	...	0.01

- **Time-Series:**

- A time-series is a sequence of values in time order
  - It is a particular case of ordered data, where order depends upon time only
  - In many applications, they consist of numerical values measured at equally spaced time intervals
    - e.g. the daily (e.g. closing) price of a company's stock in the stock market
  - In these cases, we can represent a collection of time-series in a matrix where each row is a time-series, whereas the columns represent a sequential time order. For instance:
    - **Bioinformatics:** datasets may represent the level of **expression** of **genes** over time, e.g. during some type of cellular cycle (e.g. reproduction) or exposure to stress (e.g. temperature or external substances)
- During analysis, however, it must be noticed that the **order** of the matrix columns is relevant

# Tabular Data: Matrices in R

- In R, a **matrix** is a **two-dimensional** array similar to a data frame, but unlike data frames, all the entries of a matrix must be of the **same type**. For example:

```
> mat <- matrix(data = c("x11","x21","x31", "x12","x22","x32"), nrow = 3, ncol = 2)
> mat
```

```
> mat
      [,1] [,2]
[1,] "x11" "x12"
[2,] "x21" "x22"
[3,] "x31" "x32"
```

```
> mat2 <- matrix(c("x11","x12", "x21","x22", "x31","x32"), nrow = 3, ncol = 2,
                  byrow = TRUE, dimnames = list(c("row1", "row2", "row3"), c("Col1", "Col2")))
> mat2
> mat2[3,]
> mat2["row3",]
```

```
> mat2
      Col1 Col2
row1 "x11" "x12"
row2 "x21" "x22"
row3 "x31" "x32"
```

```
> mat2[3,]
      Col1 Col2
"x31" "x32"

> mat2["row3",]
      Col1 Col2
"x31" "x32"
```

- A **vector** is analogous, but **dimensionless**
  - `c("x11","x12","x21", "x22","x31","x32")` in the codes above is a vector

# Tabular Data: Matrices in R

- If a data frame has only one type of data, we can convert it into a matrix

- Example:

```
> X <- data.frame(coll = rep(1,10), col2 = seq(from = 10, to = 55, by = 5), col3 = 5:14)
> X <- as.matrix(X)
```



```
> X
      coll col2 col3
[1,]     1   10    5
[2,]     1   15    6
[3,]     1   20    7
[4,]     1   25    8
[5,]     1   30    9
[6,]     1   35   10
[7,]     1   40   11
[8,]     1   45   12
[9,]     1   50   13
[10,]    1   55   14
```

- Column/row names (if any) can be removed if desired:

```
> dimnames(X) <- NULL
```

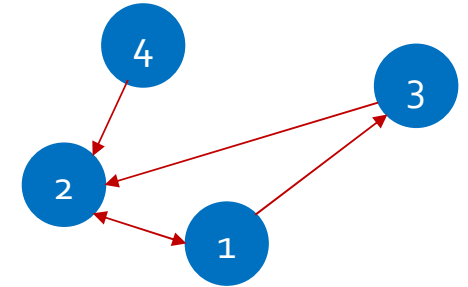
- Arithmetic/algebraic operations are supported if matrix is numeric

```
> is.numeric(X)      # TRUE
> X <- X + X          # X = 2*X
```



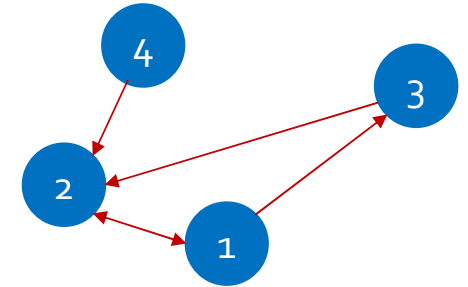
```
> X # Try it yourself!
...
```

# Note on Non-Tabular Data



- Many specialized data mining algorithms operate directly on data representations that are **not** of a tabular/matrix/vectorized nature (despite possibly being structured)
- An important example of such non-tabular data representations is **graphs**
  - e.g. a **Social/Professional** network can be modelled as a graph where nodes are **individuals** and edges are their **relationships**
  - The **WWW** can be seen as a graph where nodes are **webpages** and edges are **hyperlinks**
  - A classic example of DM on graphs is the **PageRank Algorithm**, used in the early days of Google's search engine and still used in various applications (e.g., scientific research ranks)
- Another example is databases of **varied-length time-series**

# Note on Non-Tabular Data



- Many specialized data mining algorithms operate directly on data representations that are **not** of a tabular/matrix/vectorized nature (despite possibly being structured)
- Interestingly, an important category of such data mining algorithms (most noticeably for *clustering* and *outlier detection*) only strictly require some type of *(dis)similarity measure* between relevant elements (vertices, time-series, ...) to operate
  - This ultimately comes down to a *fundamental form of tabular data for DM*: **proximity matrices**

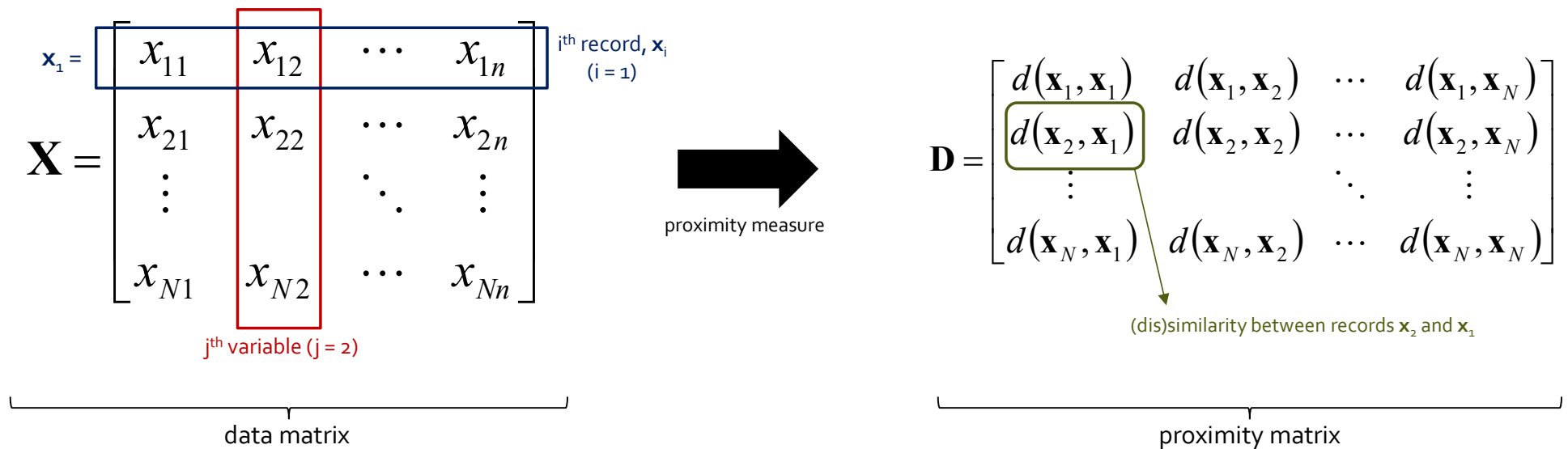


# Tabular Data: Proximity Matrices

- Some data analysis methods depend solely upon a certain measure of **proximity** between data records (customers, patients, genes, documents, etc.) to operate
  - Proximity is a measure of **similarity** or **dissimilarity**
- In other words, these methods don't need the original data matrix, but only some measure of (dis)similarity between its rows
- One can keep the original data matrix and compute (dis)similarities on demand, as the analysis being performed requires
  - The disadvantage is that the same (dis)similarity values may end up being computed several times, increasing the computational burden in terms of processing
- One alternative is to pre-compute the pairwise (dis)similarities and store them in a matrix for further use and reuse as required, without re-computations

# Tabular Data: Proximity Matrices

- For a data matrix  $\mathbf{X}$  with  $N$  data records (rows),  $\mathbf{x}_1 \dots \mathbf{x}_N$ , each of which described by  $n$  variables (columns), a **proximity matrix** is a square matrix with  $N$  rows by  $N$  columns



# Tabular Data: Proximity Matrices

$$\mathbf{D} = \begin{bmatrix} d(\mathbf{x}_1, \mathbf{x}_1) & d(\mathbf{x}_1, \mathbf{x}_2) & \cdots & d(\mathbf{x}_1, \mathbf{x}_N) \\ d(\mathbf{x}_2, \mathbf{x}_1) & d(\mathbf{x}_2, \mathbf{x}_2) & \cdots & d(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & & \ddots & \vdots \\ d(\mathbf{x}_N, \mathbf{x}_1) & d(\mathbf{x}_N, \mathbf{x}_2) & \cdots & d(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

- Besides avoiding re-computations of proximity values, using  $\mathbf{D}$  rather than  $\mathbf{X}$  has the additional advantage that the proximity matrix hides information about individual records
  - It may allow  $\mathbf{X}$  to be omitted, which can be useful for **privacy preserving** data analysis
- However, matrix  $\mathbf{D}$  may be very big and, therefore, it may turn out to be prohibitive for datasets with large amounts of records (unless it is sparse and properly stored/handled as such)
  - For instance, for a dataset approaching 1 million records, the proximity matrix would already have about 1 million x 1 million = 1 trillion entries. If each entry occupied only 1 byte (in practice it is more than that), the matrix alone would require about 1 Terabyte of memory
- Whether explicitly stored or implicitly computed on demand, proximity measures and matrices are a foundational topic in data analysis/mining and data science in general
  - For this reason, in the following we will study some of the most common measures of proximity

# PART 3

---

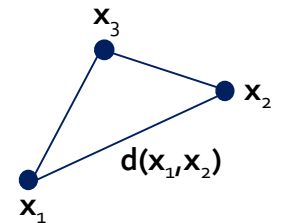
## Proximity Measures

# Proximity

- Proximity is a term that refers broadly either to **similarity** or **dissimilarity**
- **Similarity**
  - A quantitative measure,  $s(\mathbf{x}_1, \mathbf{x}_2)$ , of how much similar two observations  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are
  - The more similar the observations, the larger the similarity value
  - Usually, similarity is a value between 0 and 1, i.e.,  $s(\mathbf{x}_1, \mathbf{x}_2) \in [0, 1]$
- **Dissimilarity**
  - A quantitative measure,  $d(\mathbf{x}_1, \mathbf{x}_2)$ , of how much different two observations  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are
  - The more dissimilar the observations, the larger the dissimilarity value
  - Usually,  $d(\mathbf{x}_1, \mathbf{x}_2) \in [0, d_{\max}]$  or  $d(\mathbf{x}_1, \mathbf{x}_2) \in [0, \infty]$

# Dissimilarity: Numerical Variables

- For datasets whose observations are described by numerical variables (numerical matrices), it is common to measure dissimilarity using a **distance measure**
- Formally, in maths a distance is a dissimilarity measure that exhibits some properties:
  - $d(\mathbf{x}_1, \mathbf{x}_2) \geq 0 \quad \forall \mathbf{x}_1 \text{ and } \mathbf{x}_2$  (**non-negativity**)
  - $d(\mathbf{x}_1, \mathbf{x}_2) = 0$  if and only if  $\mathbf{x}_1 = \mathbf{x}_2$  (**identity of indiscernibles**)
  - $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1) \quad \forall \mathbf{x}_1 \text{ and } \mathbf{x}_2$  (**simmetry**)
- Besides, **d** is called a **metric** if the following also holds true:
  - $d(\mathbf{x}_1, \mathbf{x}_2) \leq d(\mathbf{x}_1, \mathbf{x}_3) + d(\mathbf{x}_3, \mathbf{x}_2) \quad \forall \mathbf{x}_1, \mathbf{x}_2, \text{ and } \mathbf{x}_3$  (**triangular inequality**)
  - This property can be very important for some data analysis/mining algorithms



# Dissimilarity: Numerical Variables

- In this case, a distance matrix will have only non-negative elements, will be symmetric, and will have only zeros in its main diagonal

$$\mathbf{D} = \begin{bmatrix} 0 & d(\mathbf{x}_1, \mathbf{x}_2) & \cdots & d(\mathbf{x}_1, \mathbf{x}_N) \\ d(\mathbf{x}_2, \mathbf{x}_1) & 0 & \cdots & d(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & & \ddots & \vdots \\ d(\mathbf{x}_N, \mathbf{x}_1) & d(\mathbf{x}_N, \mathbf{x}_2) & \cdots & 0 \end{bmatrix}$$

This part is mirrored with the bottom part in red

This part is mirrored with the upper part in green

# Dissimilarity: Numerical Variables

	Variable 1	Variable 2	Variable 3	Variable 4
$\mathbf{x}_1$	2	-1	1	0
$\mathbf{x}_2$	7	0	-4	8
$\mathbf{x}_3$	4	3	5	2
$\mathbf{x}_4$	5	10	-1	5

- The distance most commonly used in practice is the well-known **Euclidean Distance**
- Given two records,  $\mathbf{x}_i = [x_{i1} \ x_{i2} \ \dots \ x_{in}]$  and  $\mathbf{x}_j = [x_{j1} \ x_{j2} \ \dots \ x_{jn}]$ , each of which is described by  $n$  numerical variables, the Euclidean distance between them is given by:

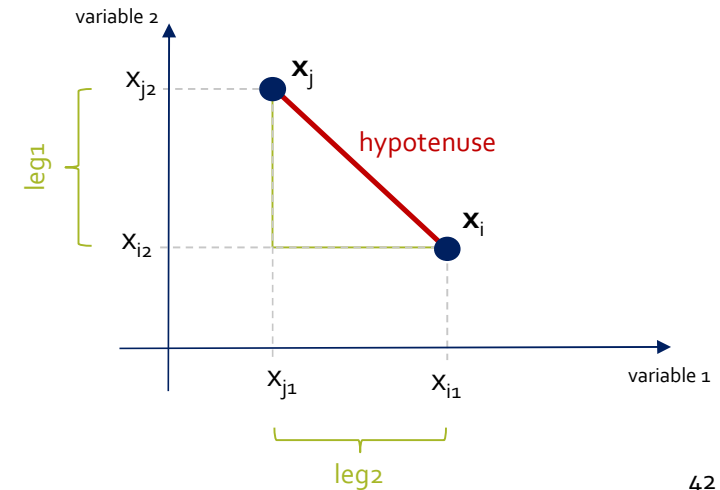
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2}$$

- Square root of the sum of the squared differences taken separately for each variable
- Example (dataset above):  $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(2 - 7)^2 + (-1 - 0)^2 + (1 - (-4))^2 + (0 - 8)^2} = \sqrt{115} = 10.72$
- **Exercise:** compute all the other distances and show the distance matrix **D** for this dataset



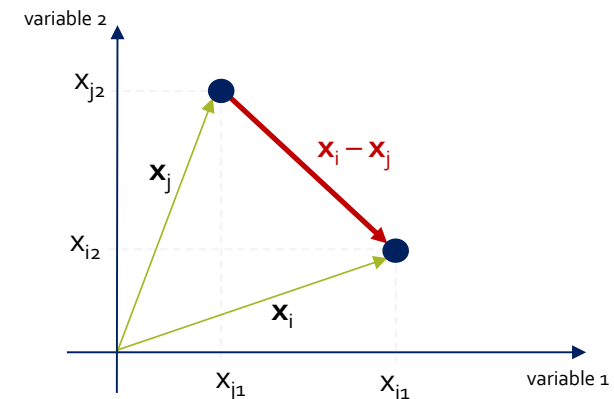
# Euclidean Distance: Geometric Interpretation 1

- We can see each record as a **point** in an n-dimensional coordinates space
- For simplicity, assume the case where we have only two variables ( $n = 2$ )
  - Two records  $\mathbf{x}_i = [x_{i1} \ x_{i2}]$  and  $\mathbf{x}_j = [x_{j1} \ x_{j2}]$  can be seen as two points on the plan
  - The Euclidean distance is the length of the line segment between them
    - It is the **hypotenuse** of a triangle
    - The legs are their differences along each coordinate
    - **Pythagoras's theorem:**
      - Hypotenuse = square root of  $((\text{leg1})^2 + (\text{leg2})^2)$



# Euclidean Distance: Geometric Interpretation 2

- We can see each record as a **vector** from the origin (0,0) to the corresponding point in an n-dimensional coordinates space
- For simplicity, assume the case where we have only two variables ( $n = 2$ )
  - Two records  $\mathbf{x}_i = [x_{i1} \ x_{i2}]$  and  $\mathbf{x}_j = [x_{j1} \ x_{j2}]$  can be seen as two vectors on the plan
  - $\mathbf{x}_i - \mathbf{x}_j$  is also a 2D vector, as shown in the figure
  - The Euclidean distance is the length of this vector
    - It is called the **magnitude** or **norm-2** of the vector
    - Designated as  $\|\mathbf{x}_i - \mathbf{x}_j\|_2$
    - Can be computed using Pythagoras's theorem as before



# Euclidean Distance in R

```
> X
      col1 col2 col3
[1,]    1  10    5
[2,]    1  15    6
[3,]    1  20    7
[4,]    1  25    8
[5,]    1  30    9
[6,]    1  35   10
[7,]    1  40   11
[8,]    1  45   12
[9,]    1  50   13
[10,]   1  55   14
```

- Let's resume one of our previous examples:

```
> X <- data.frame(col1 = rep(1,10), col2 = seq(from = 10, to = 55, by = 5), col3 = 5:14)
> X <- as.matrix(X)
```

- It is trivial to manually compute the Euclidean distance between any two rows:

- For example, 3<sup>rd</sup> and 7<sup>th</sup> rows: 

```
> d_x3x7 <- sqrt( sum( (X[3,] - X[7,])^2 ) ) # d_x3x7 = 20.39608
```

- But we can also use R function `dist`:

```
> mat_x3x7 <- rbind(X[3,], X[7,]) # Build a matrix with the 3rd and 7th rows of X
> d_x3x7 <- dist(mat_x3x7, method = "euclidean") # d_x3x7 = 20.39608
```

- If a matrix with more than two rows is given to `dist`, it returns the distance matrix `D` with all pairwise distances. For instance, try yourself the following command:

```
> D <- dist(X, method = "euclidean", diag = TRUE, upper = TRUE) # Returns the complete (Euclidean) distance matrix of X!
```

# Dissimilarity: Numerical Variables

	Variable 1	Variable 2	Variable 3	Variable 4
$\mathbf{x}_1$	2	-1	1	0
$\mathbf{x}_2$	7	0	-4	8
$\mathbf{x}_3$	4	3	5	2
$\mathbf{x}_4$	5	10	-1	5

- Another common distance is the **Manhattan Distance**
  - So-called city-block or taxicab distance

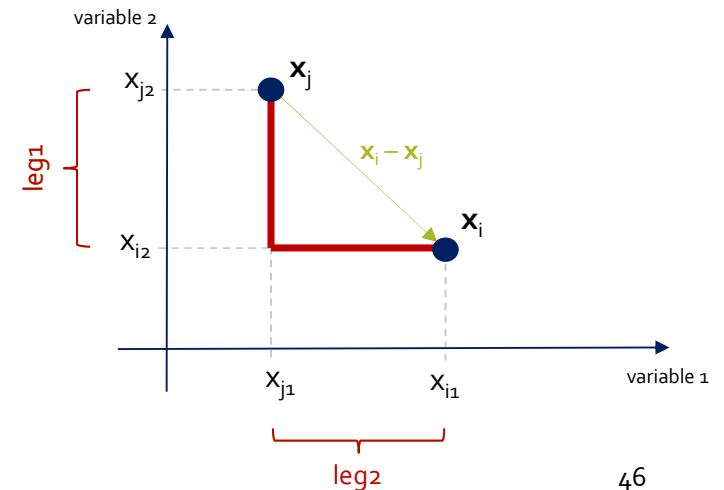
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^n |x_{ik} - x_{jk}|$$

where  $|\cdot|$  stands for absolute value (magnitude of the value regardless of its sign)

- Sum of the absolute differences taken separately for each variable
- Example (dataset above):  $d(\mathbf{x}_1, \mathbf{x}_2) = |2 - 7| + |-1 - 0| + |1 - (-4)| + |0 - 8| = 19$
- **Exercise:** compute all the other distances and show the distance matrix **D** for this dataset

# Manhattan Distance: Geometric Interpretation

- We can see each record as a **point** in an n-dimensional coordinates space
- For simplicity, assume the case where we have only two variables ( $n = 2$ )
  - Two records  $\mathbf{x}_i = [x_{i1} \ x_{i2}]$  and  $\mathbf{x}_j = [x_{j1} \ x_{j2}]$  can be seen as two points on the plan
  - The Manhattan distance is the sum of the legs of a right triangle with vertices  $\mathbf{x}_i$  and  $\mathbf{x}_j$
  - Name comes from this interpretation:
    - "To go from  $\mathbf{x}_i$  to  $\mathbf{x}_j$ , go 'around the corner' "
- Mathematically, it is also the **norm-1** of **vector**  $\mathbf{x}_i - \mathbf{x}_j$ 
  - $\|\mathbf{x}_i - \mathbf{x}_j\|_1 = \text{leg1} + \text{leg2}$



# Dissimilarity: Numerical Variables

	Variable 1	Variable 2	Variable 3	Variable 4
$\mathbf{x}_1$	2	-1	1	0
$\mathbf{x}_2$	7	0	-4	8
$\mathbf{x}_3$	4	3	5	2
$\mathbf{x}_4$	5	10	-1	5

- Both the Euclidean and Manhattan distances are particular cases of a more general, parameterised distance known as **Minkowski Distance**

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p = \left( \sum_{k=1}^n |x_{ik} - x_{jk}|^p \right)^{1/p}$$

- By setting parameter p to different values we get different distances, for instance:
  - p = 1 (it is trivial to see that we get the **Manhattan** distance)
  - p = 2 (it is trivial to see that we get the **Euclidean** distance)
  - p  $\rightarrow \infty$  (it can be shown that as p tends to infinity, in the limit we get the **Suprema** distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_\infty = \max_{1 \leq k \leq n} |x_{ik} - x_{jk}| \quad \Rightarrow \quad \text{e.g. } d(\mathbf{x}_1, \mathbf{x}_2) = \max(|2 - 7|, |-1 - 0|, |1 - (-4)|, |0 - 8|) = 8 \text{ (example above)}$$

# Minkowski Distance in R

```
> X
      col1 col2 col3
[1,]    1  10    5
[2,]    1  15    6
[3,]    1  20    7
[4,]    1  25    8
[5,]    1  30    9
[6,]    1  35   10
[7,]    1  40   11
[8,]    1  45   12
[9,]    1  50   13
[10,]   1  55   14
```

- Let's resume one of our previous examples:

```
> X <- data.frame(col1 = rep(1,10), col2 = seq(from = 10, to = 55, by = 5), col3 = 5:14)
> X <- as.matrix(X)
```

- It is trivial to manually compute the Minkowski distance between any two rows:

- For example, 3<sup>rd</sup> and 7<sup>th</sup> rows:

```
> d_x3x7 <- sum( abs(X[3,] - X[7,]) ) # d_x3x7 = 24 (Manhattan)
> d_x3x7 <- max( abs(X[3,] - X[7,]) ) # d_x3x7 = 20 (Suprema)
```

- But we can also use R function `dist`:

```
> mat_x3x7 <- rbind(X[3,], X[7,]) # Build a matrix with the 3rd and 7th rows of X
> d_x3x7 <- dist(mat_x3x7, method = "minkowski", p = 1) # d_x3x7 = 24 (Manhattan)
> d_x3x7 <- dist(mat_x3x7, method = "maximum") # d_x3x7 = 20 (Suprema)
```

- Or for the entire data matrix:

```
> D <- dist(X, method = "manhattan", diag = TRUE, upper = TRUE) # Returns the complete (Manhattan) distance matrix of X
> D <- dist(X, method = "maximum", diag = TRUE, upper = TRUE) # Returns the complete (Suprema) distance matrix of X
```

# Exercises (Minkowski)

- **Exercise 1:** Given a fixed reference point  $\mathbf{x}_1$  on the plane, and another point  $\mathbf{x}_2$  at a distance  $d(\mathbf{x}_1, \mathbf{x}_2)$  from  $\mathbf{x}_1$ , it is easy to see that the set of all points that are at this very same distance from  $\mathbf{x}_1$  form a circle with radius  $d(\mathbf{x}_1, \mathbf{x}_2)$  centred at  $\mathbf{x}_1$  if  $d$  is the **Euclidean** distance. What are the shapes formed if  $d$  is: **Manhattan**? **Suprema**?
- **Exercise 2:**

Given two records  $\mathbf{p}$  and  $\mathbf{q}$  each described by  $n=6$  numerical variables as following:

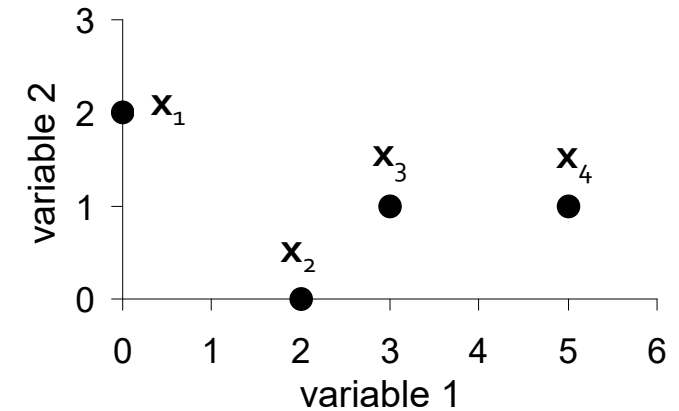
$$\begin{aligned}\mathbf{p} &= [1 \quad 2 \quad -3 \quad 2 \quad 0 \quad 8] \\ \mathbf{q} &= [0 \quad 6 \quad 2 \quad -1 \quad 2 \quad 5]\end{aligned}$$

Compute the Euclidean, Manhattan, and Suprema distances: (a) manually; (b) using R



# Exercises (Minkowski)

- **Exercise 3:** Compute the distance matrices for the following data set using **Euclidean**, **Manhattan**, and **Suprema** distances: (a) Manually; and (b) Using R



	variable 1	variable 2
x1	0	2
x2	2	0
x3	3	1
x4	5	1

Distance Matrix

	x1	x2	x3	x4
x1	0	???	???	???
x2	???	0	???	???
x3	???	???	0	???
x4	???	???	???	0

# Real-World Example: Iris Dataset



Iris Setosa

CC BY-SA 3.0,

<https://commons.wikimedia.org/w/index.php?curid=170298>



Iris Versicolor

CC BY-SA 3.0,

<https://commons.wikimedia.org/w/index.php?curid=248095>



Iris Virginica

By Frank Mayfield, CC BY-SA 2.0,

<https://commons.wikimedia.org/w/index.php?curid=9805580>

- Very famous dataset introduced in 1936 by Ronald Fisher
  - 150 observations of flowers:
    - 50 observations from each of three species of Iris: Setosa, Virginica, Versicolor
    - Four variables: the length and the width of the sepals and petals, in centimetres
  - Dataset is available, for instance, at the UCI (University of California – Irvine) repository:
    - CSV file containing no variable names (no header) and the Iris category as a fifth variable (column)
  - Reading the file and computing the (Euclidean) distance matrix using R:

```
...  
5.1,3.8,1.6,0.2,Iris-setosa  
4.6,3.2,1.4,0.2,Iris-setosa  
5.3,3.7,1.5,0.2,Iris-setosa  
5.0,3.3,1.4,0.2,Iris-setosa  
7.0,3.2,4.7,1.4,Iris-versicolor  
6.4,3.2,4.5,1.5,Iris-versicolor  
6.9,3.1,4.9,1.5,Iris-versicolor  
...
```

Fraction of the Iris CSV file

```
> Iris <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data",  
                    header = FALSE, sep = ",", dec = ".") # Same as read.csv()  
> Iris <- as.matrix(Iris[,1:4]) # Convert data.frame to matrix...  
                                # ... getting rid of the last (categorical class) variable  
> dist(Iris, method = "euclidean")
```

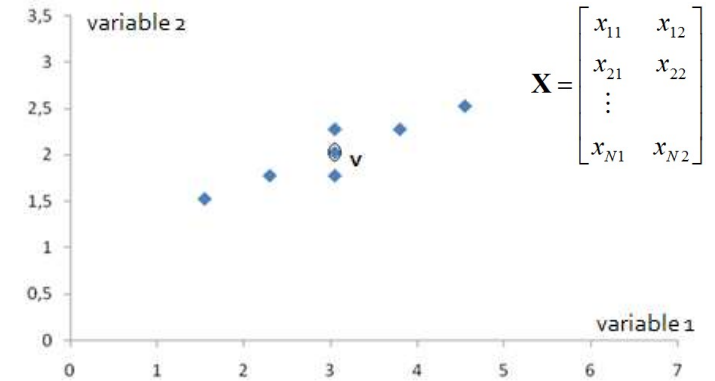
# Exercise (Minkowski in R)

- Ionosphere Radar Data
  - see description at <https://archive.ics.uci.edu/ml/datasets/Ionosphere>
- The dataset is also available at the UCI repository as a CSV file :
  - <https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data>
    - CSV file with no variable names (no header), "." as decimal point and "," as separator
    - 351 records (rows) described by 34 numeric variables (columns)
    - 35<sup>th</sup> column is a categorical class label ("good" or "bad")
  - **Exercise:** Read the file into a data frame in R, convert it to a matrix getting rid of the class label (last column), then compute the Euclidean, Manhattan and Suprema distance matrices for the submatrix containing the first 10 records of the data only

# Dissimilarity: Numerical Variables

- Another important distance in multivariate analysis is the **Mahalanobis Distance**
- Originally, and most commonly, this is a distance between a point  $\mathbf{x}$  and a cloud of points (or cluster) as represented by its centre (multivariate mean),  $\mathbf{v}$
- Generally speaking, the Mahalanobis distance is a generalisation of the Euclidean distance that takes into account the covariance of the variables as measured from a cloud of points around their centre  $\mathbf{v}$
- In the following we discuss the intuition and the computation of this distance in R
  - All we need are basic definitions of covariance and multivariate mean

# Mahalanobis Distance



- **Multivariate mean (data centre):**

- Given a collection  $X$  of  $N$  points,  $\mathbf{x}_1 \dots \mathbf{x}_N$ , with coordinates  $x_{11} \dots x_{N1}$  for variable 1 and  $x_{12} \dots x_{N2}$  for variable 2, respectively, the **centre** or **multivariate mean** of  $X$  (so-called the **centroid** of  $X$ ) is a point  $\mathbf{v} = [v_1 \ v_2]$  whose coordinates  $v_1$  and  $v_2$  are the mean values of  $X$  in the corresponding variables, i.e.,  $v_1 = (x_{11} + x_{21} + \dots + x_{N1})/N$  and  $v_2 = (x_{12} + x_{22} + \dots + x_{N2})/N$  (function `mean()` in R)

- **Covariance:**

- The (sample) covariance between variables 1 and 2 for the cloud of points in  $X$  is defined as:

$$\text{cov}(\text{variable 1, variable 2}) = \frac{(x_{11} - v_1) \cdot (x_{12} - v_2) + \dots + (x_{N1} - v_1) \cdot (x_{N2} - v_2)}{N - 1} = \frac{1}{N - 1} \sum_{k=1}^N (x_{k1} - v_1)(x_{k2} - v_2)$$

- **Intuition:** if higher (lower) values in one variable are associated with higher (lower) values of the other, the products will be maximized. So, cov is high and positive (negative) when the variables are positively (negatively) correlated. If they are not correlated, the products tend to cancel out (cov  $\approx$  0)

# Mahalanobis Distance

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix} \Rightarrow \text{Cov}(\mathbf{X}) = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

- **Covariance Matrix:**

- In general,  $\mathbf{X}$  may have  $n$  variables (points with  $n$  coordinates)
- The covariance matrix of a set of data points  $\mathbf{X}$  is a matrix where the element in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column,  $c_{ij}$ , is the covariance between variables  $i$  and  $j$ , i.e.,  $c_{ij} = \text{cov}(\text{variable } i, \text{variable } j)$
- It is a squared ( $n \times n$ ) matrix whose elements in the diagonal are the **variances** of the variables

$$c_{ii} = \text{cov}(\text{variable } i, \text{variable } i) = \frac{(x_{1i} - v_i) \cdot (x_{1i} - v_i) + \cdots + (x_{Ni} - v_i) \cdot (x_{Ni} - v_i)}{N - 1} = \frac{1}{N - 1} \sum_{k=1}^N (x_{ki} - v_i)^2 = \sigma_i^2$$

- The variance  $(\sigma_i)^2$  as defined above is the square of the **sample** standard deviation  $\sigma_i$
- In **R**, the standard deviation, variance, and covariance matrix are readily computed by functions `sd()`, `var()`, and `cov()`

# Mahalanobis Distance

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix} \rightarrow \text{Cov}(\mathbf{X}) = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

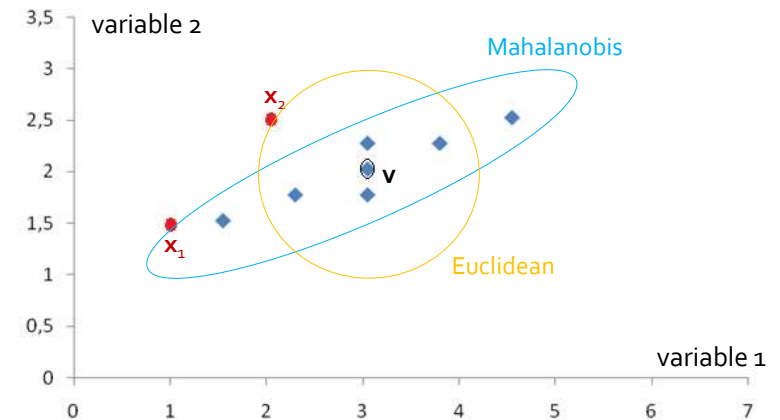
- The Mahalanobis distance between a point  $\mathbf{x}$  and the centre  $\mathbf{v}$  of a set of data  $\mathbf{X}$  is:

$$d(\mathbf{x}, \mathbf{v}) = \sqrt{(\mathbf{x} - \mathbf{v})^T \text{Cov}(\mathbf{X})^{-1} (\mathbf{x} - \mathbf{v})}$$

- Geometric Interpretation:
  - Basically, Euclidean distance on a PCA transformed representation of the dataset
  - Formal Analysis: M. J. Zaki and W. Meira Jr., "Data Mining and Machine Learning. Fundamental Concepts and Algorithms", 2nd Ed., Cambridge Univ. Press, 2020 (Section 2.5)

# Mahalanobis Distance

- Example 1:
  - 2-dimensional dataset with 7 points (blue diamonds)
  - The blue points suggest a high correlation between variables 1 and 2
    - Higher (lower) values of var. 1 are associated with higher (lower) values of var. 2 and vice-versa
    - This can make the Mahalanobis distance from the centre  $\mathbf{v}$  of the cloud of points to an external point  $\mathbf{x}_1$  shown in the figure to be smaller than that to another point  $\mathbf{x}_2$ , i.e.,  $d(\mathbf{x}_1, \mathbf{v}) < d(\mathbf{x}_2, \mathbf{v})$ 
      - Visually, we can see that the Euclidean distance from  $\mathbf{v}$  to  $\mathbf{x}_2$  is actually smaller than that from  $\mathbf{v}$  to  $\mathbf{x}_1$
      - However, by taking the covariance into account, the **Mahalanobis** distance induces **ellipsoidal** (rather than spherical) equidistant point sets, which makes  $\mathbf{v}$  closer to  $\mathbf{x}_1$  than it is to  $\mathbf{x}_2$

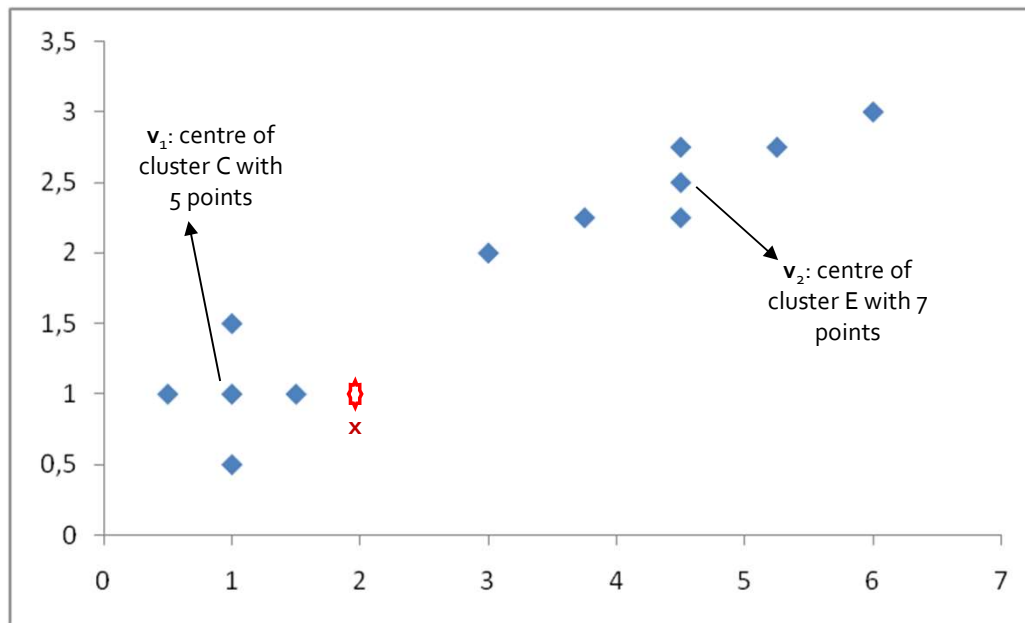




# Mahalanobis Distance

- Example 2:

```
C <- matrix(c(0.5, 1, 1, 0.5, 1, 1, 1, 1.5, 1.5, 1), 5, 2, byrow = TRUE) # Cluster C
v1 <- c(mean(C[,1]), mean(C[,2])) # Compute Centre of Cluster C (2-dimensional mean)
cov_C <- cov(C) # Compute Covariance Matrix of Cluster C
x <- c(2,1) # Point x
dxv1_sq <- mahalanobis(x, v1, cov_C) # [SQUARED] Mahalanobis dist. from x to v1
dxv1 <- sqrt(dxv1_sq) # Mahalanobis dist. from x to v1
```



- With the R code above we can compute the Mahalanobis distance from  $\mathbf{x} = [2 \ 1]$  to  $\mathbf{v}_1 = [1 \ 1]$  as the centre of cluster C (with five points, at the bottom left corner of the figure)
- The resulting distance is  $d(\mathbf{x}, \mathbf{v}_1) = 2.82$  ( $d(\mathbf{x}, \mathbf{v}_1)^2 = 8$ )
- **Exercise:** If the covariance matrix of cluster E (with 7 points at the upper right corner of the figure) is:

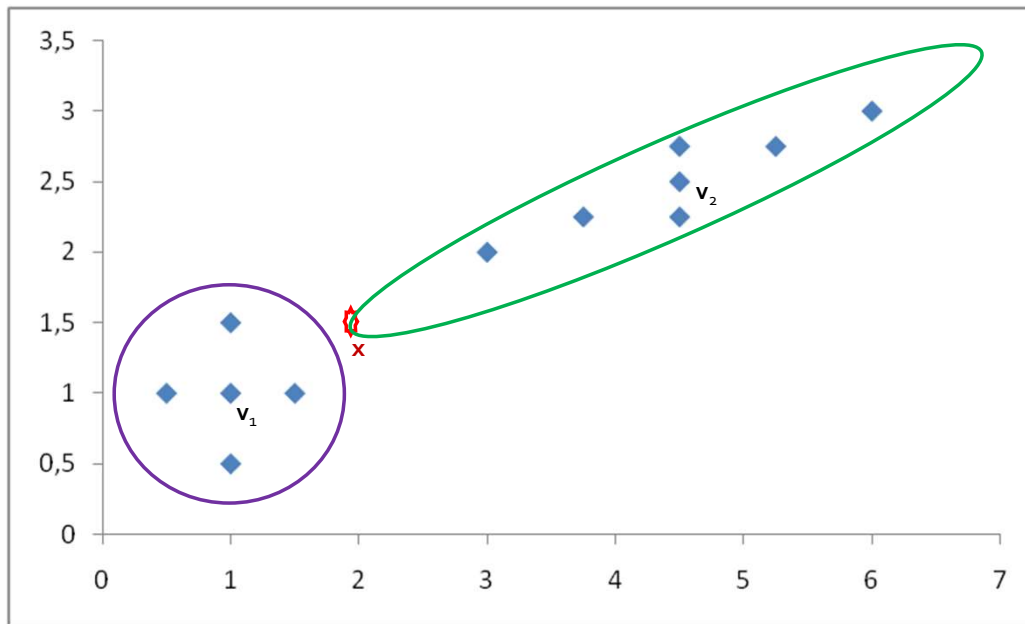
$$\text{Cov}_E = \begin{bmatrix} 0.80 & 0.27 \\ 0.27 & 0.11 \end{bmatrix}$$

and the centre is  $\mathbf{v}_2 = [4.5 \ 2.5]$ , compute the distance from  $\mathbf{x}$  to  $\mathbf{v}_2$ ,  $d(\mathbf{x}, \mathbf{v}_2)$ , using function `mahalanobis()` in R\*

- The result of the exercise above should be  $d(\mathbf{x}, \mathbf{v}_2)^2 = 30.62$  (what the function returns), i.e.,  $d(\mathbf{x}, \mathbf{v}_2) = 5.53$
- Notice that, in this case,  $\mathbf{x}$  is much closer to Cluster C than it is to Cluster E
- **Exercise:** Repeat the computations now using the formula directly, rather than the `mahalanobis()` function

# Mahalanobis Distance

- Example 2 (continued):



- Now we have exactly the same scenario as before, except that we have moved point  $x$  slightly to position  $[2 \ 1.5]$
- **Exercise:** Re-compute distances  $d(x, v_1)$  and  $d(x, v_2)$ , noticing that only point  $x$  and the Mahalanobis distances themselves need to be re-computed, nothing else
- The result of the above exercise should be  $d(x, v_1) = 3.16$  and  $d(x, v_2) = 3.01$ . That is,  $x$  is now closer to  $v_2$  (Ellipsoidal Cluster E) than to  $v_1$  (Spherical Cluster C) !

# Note on Variable Scales

- Distances such as Minkowski and Mahalanobis can be affected by the scales of the variables in a way that their relative differences can be altered if scales are changed
- Applications such as data clustering generally **will not be scale invariant** when these distances are used
- Variables ranging within wider intervals tend to dominate the results

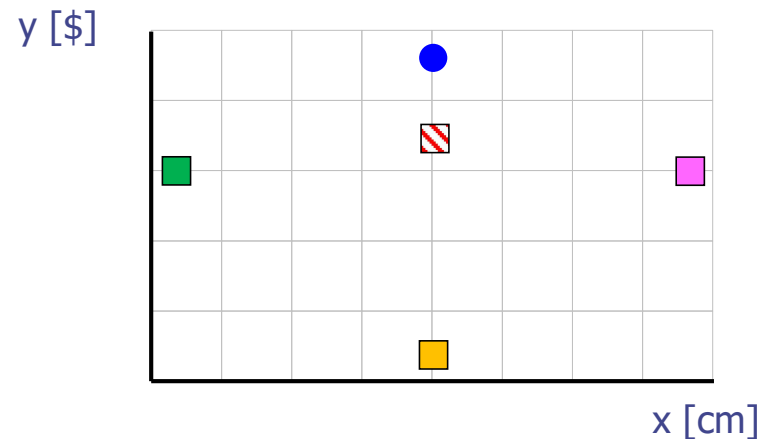
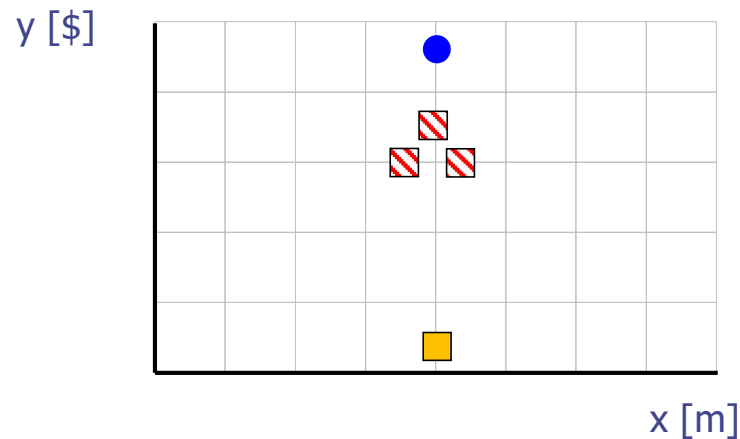
# Note on Variable Scales

- Example (Euclidean Distance):

$$d^E(\mathbf{x}_1, \mathbf{x}_2) = ?$$

	<b>a<sub>1</sub></b>	<b>a<sub>2</sub></b>	<b>a<sub>3</sub></b>	<b>a<sub>4</sub></b>
<b>x<sub>1</sub></b>	1	2	5	803
<b>x<sub>2</sub></b>	1	1	5	712
<b>x<sub>3</sub></b>	1	1	5	792
<b>x<sub>4</sub></b>	0	2	6	608
<b>x<sub>5</sub></b>	0	1	5	677
<b>x<sub>6</sub></b>	1	1	5	927
<b>x<sub>7</sub></b>	1	1	5	412
<b>x<sub>8</sub></b>	1	1	6	368
<b>x<sub>9</sub></b>	1	1	6	167
<b>x<sub>10</sub></b>	0	2	5	847
<b>Mean</b>	<b>0,70</b>	<b>1,30</b>	<b>5,30</b>	<b>631,30</b>
<b>Variance</b>	<b>0,23</b>	<b>0,23</b>	<b>0,23</b>	<b>59045,34</b>

# Note on Variable Scales



- Different scales may be just an artefact resulting from the use different unit measurements...
- This can be easily addressed by variable **rescaling** or **standardisation**
- The two simplest and most common techniques are *linear rescaling* and *z-score standardisation*

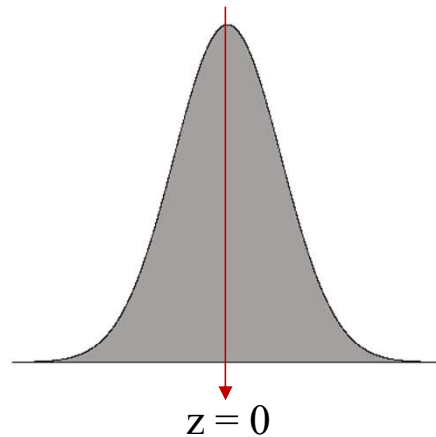
## Rescaling / Standardisation

- Linear Rescaling [0,1]: 
$$l_{ij} = \frac{x_{ij} - \min(\mathbf{a}_j)}{\max(\mathbf{a}_j) - \min(\mathbf{a}_j)}$$

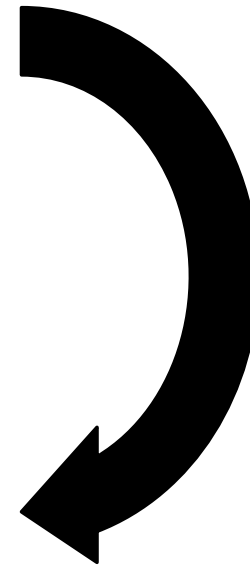
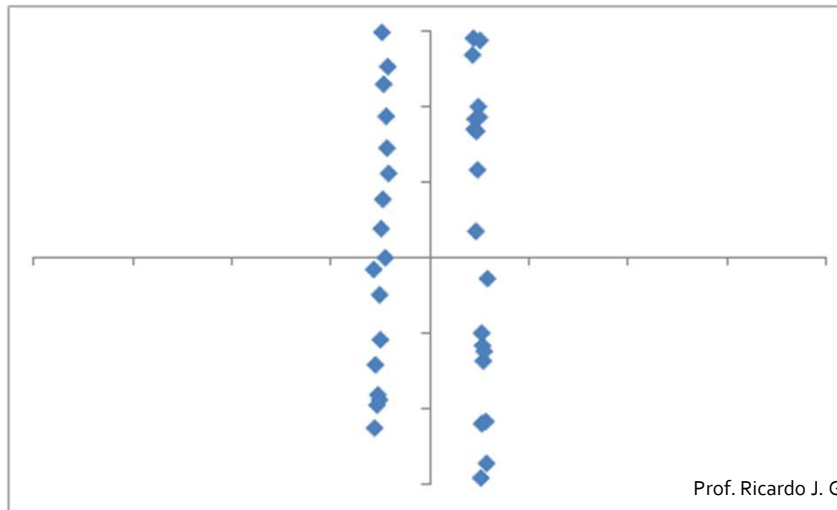
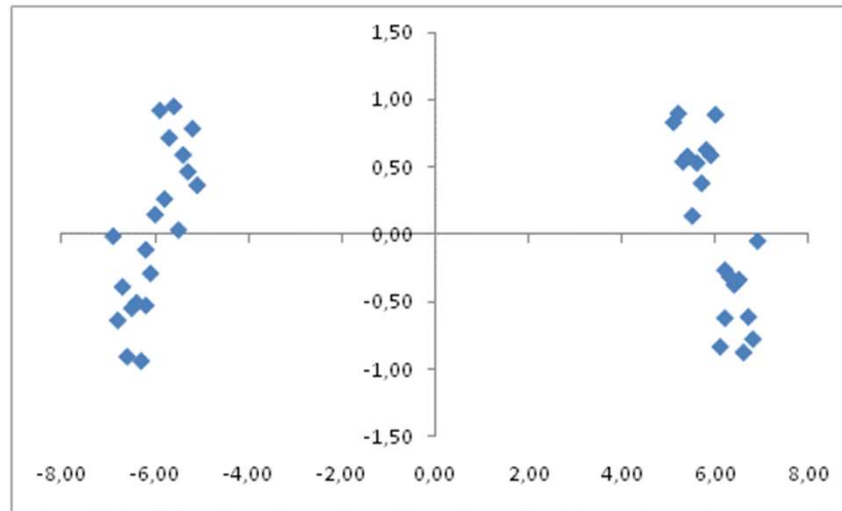
- z-Score Standardisation: 
$$z_{ij} = \frac{x_{ij} - \mu_{\mathbf{a}_j}}{\sigma_{\mathbf{a}_j}}$$



N(0,1) if variable  $\mathbf{a}_j$  follows  
a Normal distribution



Variable rescaling / standardisation is not necessarily always helpful / useful ...



*z-score std.*  
(similar for linear  
rescaling [0,1])

# Note on Variable Scales

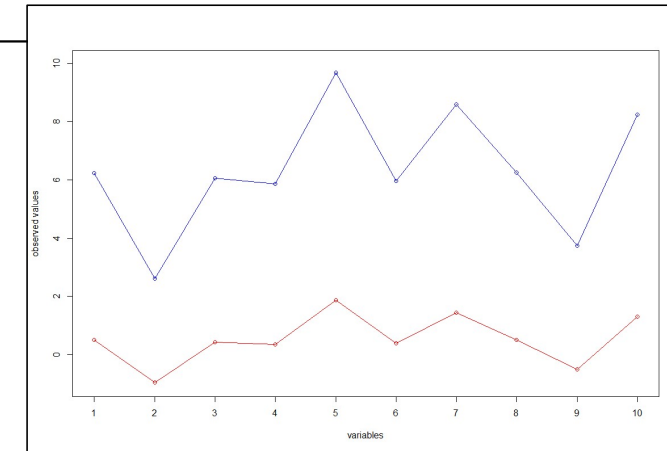
- ❑ Variables ranging within wider scales tend to dominate computations of certain distances measures
  - this can be seen as a type of implicit pre-weighting of the variables
- Re-scaling / standardisation removes this pre-weighting, assumed to be artificial or undesirable
  - e.g., due to the use of different unit measurements
  - this is equivalent to impose a (counter) weighing under the assumption that all variables should weigh the same
  - as such, it can introduce distortions if the original variability at least partially reflected the nature of the problem
  - it can significantly change the results of certain types of data analyses, e.g., clustering, outlier detection, PCA, ...



# Note on Variable Scales

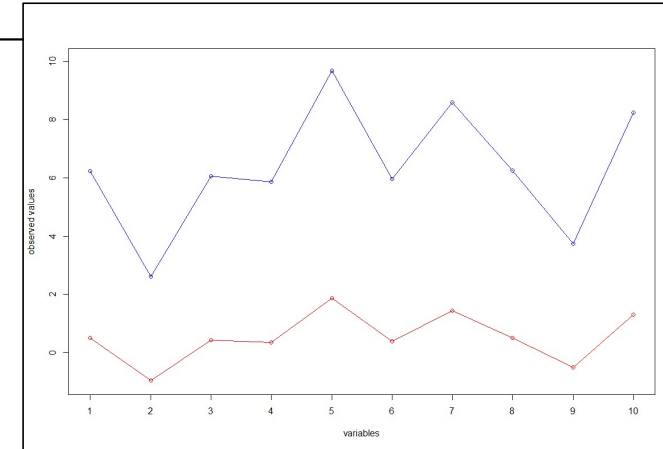
- It is difficult to make general recommendations, which are not domain-specific or problem dependent
- Variable pre-processing is part of the **cycle** of KDD
- Such KDD cycle should be approached as such, i.e., from an **Exploratory Data Analysis** perspective!

# Similarity: Numerical Variables



- Distances as those discussed before are widely used in applications involving numerical data, but they are not always an appropriate choice
- In some application domains the notion of proximity between observations is better captured by the “trend” of their variables rather than by their absolute values
- A classic example is **gene-expression** data in bioinformatics
  - Genes are observations and their expression levels at different experimental conditions are the variables. In this domain, genes may have similar expression profiles (be “co-regulated”) if their expression levels are somehow **correlated**, rather than similar in magnitude
- Consider the example in the figure above, where  $n = 10$  variables are represented along the x axis and their values for 2 observations ( $x_i$  in red and  $x_j$  in blue) are in the y axis

# Similarity: Numerical Variables

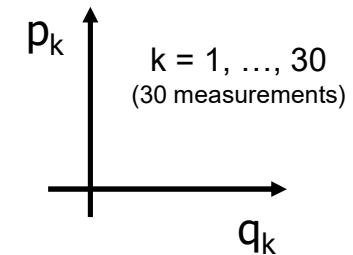
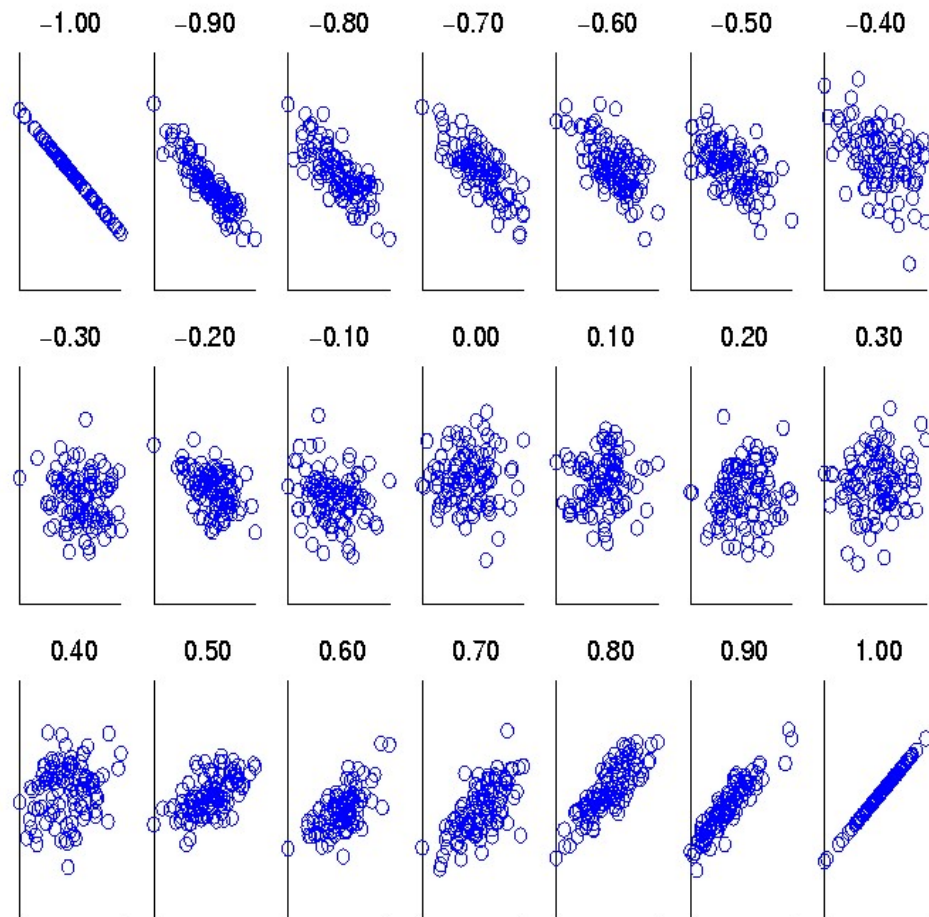


- Notice that the magnitude of the values are very different, but the trend is the same
  - The 2 observations are highly correlated across their values for the different variables
  - Actually, the observation in blue is obtained by multiplying the values in the red one by 2.5 and then adding a constant equal to 5 (i.e.  $x_{jk} = 2.5 \cdot x_{ik} + 5$  for all  $k = 1, 2, \dots, 10$ )
  - In other words, these two observations are perfectly linearly correlated
- The degree of correlation can be measured by the **Pearson measure**: 
$$r(\mathbf{x}_i, \mathbf{x}_j) = \frac{\text{cov}(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_{\mathbf{x}_i} \cdot \sigma_{\mathbf{x}_j}}$$
where cov and  $\sigma$  denote covariance and standard deviation exactly as previously defined, but here computed for observations across their values for the different variables, not the opposite

# Pearson Correlation

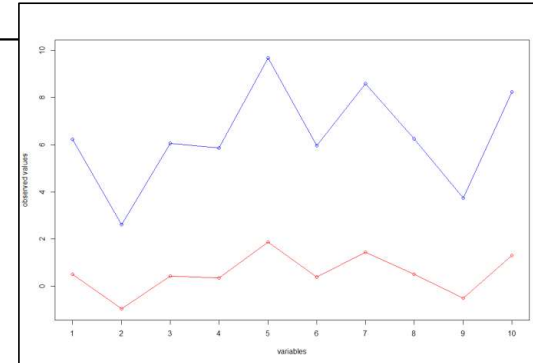
- Pearson is a **similarity measure** because the higher its value the higher the similarity
- It can be used to measure correlation between two observations across their descriptive variables, as in the previous example, but it can also be used to measure correlation between two numerical variables across a set of observations
- Pearson captures linear correlation in the interval  $[-1, +1]$ 
  - $r(\mathbf{p}, \mathbf{q}) = +1$ : observations or variables  $\mathbf{p}$  and  $\mathbf{q}$  are perfectly positively correlated (same trend)
  - $r(\mathbf{p}, \mathbf{q}) = -1$ :  $\mathbf{p}$  and  $\mathbf{q}$  are perfectly negatively correlated (opposite trend)
  - $r(\mathbf{p}, \mathbf{q}) = 0$ :  $\mathbf{p}$  and  $\mathbf{q}$  are not correlated at all
    - this is the expected values when  $\mathbf{p}$  and  $\mathbf{q}$  are statistically independent from each other
- It can be better visualized by a scatter plot where the  $k^{\text{th}}$  values of  $\mathbf{p}$  and  $\mathbf{q}$ ,  $p_k$  and  $q_k$  respectively, are taken and plotted as a point  $(p_k, q_k)$  on the plan, for all measurements  $k$

# Visually Evaluating Correlation



**Scatter plots  
showing the  
similarity from  
-1 to 1**

# Pearson Correlation



- Our 1<sup>st</sup> example (figure reproduced above) shows a scenario where the **Euclidean** distance would suggest that the observations in red and blue are far from each other if seen as two **points** in a 10-dimensional coordinates space
  - Differently, **Pearson** captures a perfect linear correlation between them as a **trend**
- But in spite of being widely used in practice, Pearson has its shortcomings
  - A well-known issue is that the measure is sensitive to extreme values, called outliers
  - For instance, consider the correlation between  $\mathbf{p} = [1 \ -3 \ 0 \ 4 \ 1 \ 0 \ 3]$  and  $\mathbf{q} = [1 \ 1 \ 4 \ -2 \ 3 \ -1 \ 4]$ 
    - It is simple to show that  $r(\mathbf{p}, \mathbf{q}) = -0.1107$  (**Exercise**: show it doing the calculations manually), i.e., pretty low
  - By changing only the first value of both  $\mathbf{p}$  and  $\mathbf{q}$  from 1 to 10,  $r(\mathbf{p}, \mathbf{q})$  increases to 0.6387!
    - **Exercise**: show it doing the calculations manually

# Spearman Correlation

```
# Computing Pearson and Spearman correlations in R:  
p <- c(1, -3, 0, 4, 1, 0, 3)  
q <- c(1, 1, 4, -2, 3, -1, 4)  
cor(p, q, method = "pearson") # Same as cov(p,q)/(sd(p)*sd(q))  
cor(p, q, method = "spearman")
```

- There exist other correlation measures, some of which are more robust to outliers than Pearson. An exhaustive discussion is not doable, so we limit ourselves here to one of the most well-known alternatives to Pearson, the **Spearman correlation**
- **Spearman:** In order to avoid sensitivity to outliers, the Spearman correlation ignores the magnitudes and uses solely the ranks of the values in question. In other words, it is equivalent to replacing the values with their relative ranks, then computing Pearson
- **Example 1:** Consider again  $\mathbf{p} = [1 \ -3 \ 0 \ 4 \ 1 \ 0 \ 3]$  and  $\mathbf{q} = [1 \ 1 \ 4 \ -2 \ 3 \ -1 \ 4]$ . Replacing the values in  $\mathbf{p}$  and  $\mathbf{q}$  with their relative ranks (intermediate rank is taken in case of ties) yields:
  - $\mathbf{p}_r = [3.5 \ 7 \ 5.5 \ 1 \ 3.5 \ 5.5 \ 2]$  and  $\mathbf{q}_m = [4.5 \ 4.5 \ 1.5 \ 7 \ 3 \ 6 \ 1.5]$  and computing Pearson gives -0.1111
    - This means that the Spearman correlation between  $\mathbf{p}$  and  $\mathbf{q}$  is  $r_s(\mathbf{p}, \mathbf{q}) = -0.1111$  (recall that  $r(\mathbf{p}, \mathbf{q}) = -0.1107$ )
  - Now replacing the first value of  $\mathbf{p}$  and  $\mathbf{q}$  with 10 as before results in  $r_s(\mathbf{p}, \mathbf{q}) = 0.3$  (against  $r(\mathbf{p}, \mathbf{q}) = 0.6387$ )
  - **Exercise:** Compute the values above manually as well as using function `cor()` in R (right top corner)

# Similarity: Numerical Variables

- There are scenarios in which neither distances nor correlations properly capture the desired notion of proximity. One such example is text, as described before, using a bag-of-words (BOW) representation
- Recall that in this kind of (BOW) representation a text document is essentially described based on the number of times that each relevant term appear in the document.
- However, since each document usually contains only a small fraction of the terms that could ever be present in the text, the representation tends to be sparse, i.e., documents tend to be described as records with many variables equal to zero
- As discussed before, these variables are **asymmetric**: documents are (dis)similar because of the terms that they have, not because of the terms they do not have
  - In other words, zeros (absent terms) should not affect the similarity between documents
- Zeros affect both distances and correlation, so something else is needed



# Similarity: Numerical Variables

- A similarity measure that is widely used in applications involving asymmetric numerical variables, such as BOW, is the **Cosine similarity**
- The Cosine similarity between two numerical observations  $\mathbf{x}_j$  and  $\mathbf{x}_i$  is given by:

$$\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{x}_j\|_2}$$

where  $\|\mathbf{x}_i\|_2$  is the magnitude of a vector (as previously defined when discussing Euclidean distance) and  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  is the inner product of two vectors, given by the sum of the individual products between their elements:

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^n x_{ik} x_{jk}$$

# Cosine Similarity

```
# Computing Cosine Similarity in R:
p <- c(3, 2, 0, 5, 0, 0, 0, 2, 0, 0)
q <- c(1, 0, 0, 0, 0, 0, 0, 1, 0, 2)
cos_pq <- (p%*%q)/sqrt((p%*%p)*(q%*%q))

# Or Alternatively:
cos_pq <- sum(p*q)/sqrt(sum(p^2)*sum(q^2))
```

- Example:

- $\mathbf{p} = [3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0]$

- $\mathbf{q} = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2]$

- $\cos(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle / \|\mathbf{p}\|_2 \|\mathbf{q}\|_2$

- $\langle \mathbf{p}, \mathbf{q} \rangle = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

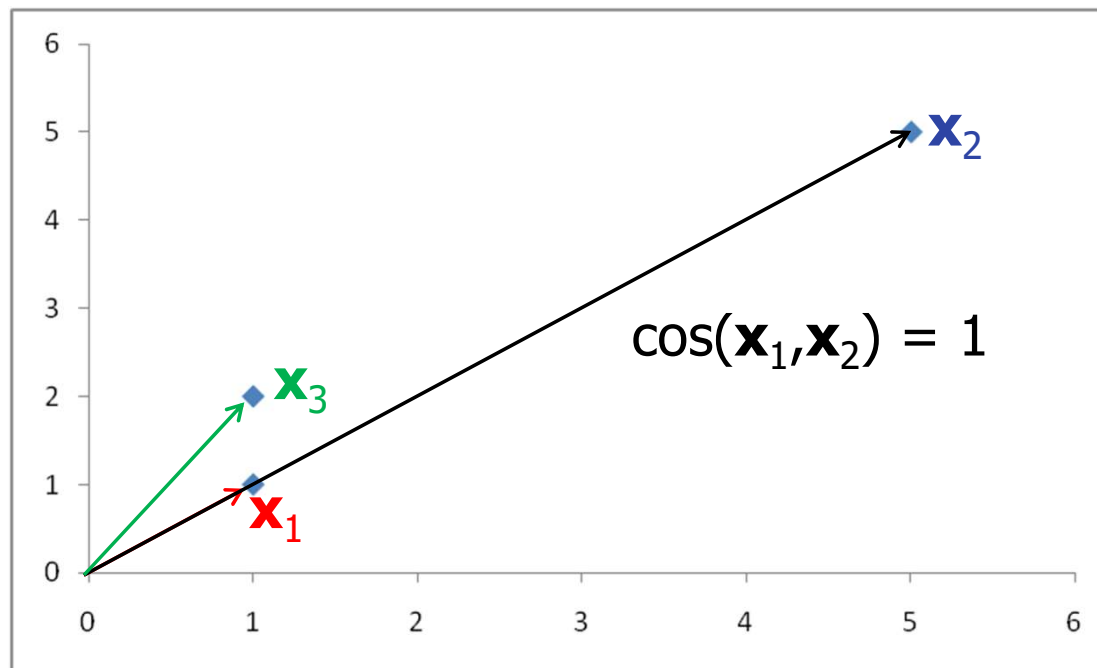
- $\|\mathbf{p}\|_2 = (3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)^{1/2} = \sqrt{42} = 6.481$

- $\|\mathbf{q}\|_2 = (1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2)^{1/2} = \sqrt{6} = 2.245$

- $\cos(\mathbf{p}, \mathbf{q}) = 0.3149$

- See text box at the top right corner for code in R

# Cosine Similarity: Geometric Interpretation



- $\cos(\mathbf{x}_1, \mathbf{x}_3) = \cos(\mathbf{x}_2, \mathbf{x}_3) = 0.95$  (angle  $\approx 18^\circ$ )

- We can see each data record as a **vector** from the origin (0,0) to the corresponding point in an n-dimensional coordinates space
- It can be shown that the cosine similarity is the cosine of the angle between these vectors
- If the variables take on non-negative values only, which is the case of bag-of-words for text,  $\cos \in [0,1]$  (angle  $\in [-90^\circ, 90^\circ]$ )
- If a document ( $\mathbf{x}_1$ ) is appended to itself several times, this increases the counts of each term in the resulting document ( $\mathbf{x}_2$ ), but doesn't affect their proportions
  - $\cos(\mathbf{x}_1, \mathbf{x}_2) = 1$  will still correctly suggest that they are equivalent

# Similarity: Cosine

- **Exercise:**

- Compute the cosine similarity between the following records:

- $\mathbf{p} = [1 \ 0 \ 0 \ 4 \ 1 \ 0 \ 0 \ 3]$

- $\mathbf{q} = [0 \ 5 \ 0 \ 2 \ 3 \ 1 \ 0 \ 4]$

- (a) Manually
- (b) In R

# (Dis)similarity: Categorical Variables

- The proximity measures for numerical variables that we have seen so far do not suit categorical data such as in the example beside
  - They would demand the categorical variables to be converted to numerical ones (discussed later), which may not always be desirable/ideal
- There are (dis)similarity measures particularly suitable for categorical data
  - In the following we discuss some of the most commonly used such measures

Weather Data from (Witten et al. 2016)

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

# Binary Variables

- We start our discussion with **binary variables**
  - e.g. Windy (False/True) and Play (No/Yes) in the “Weather data”
  - Notice that, for the sake of simplicity, we can standardise the representation of the two values of any binary categorical variable as “0” (e.g. for False, No) and “1” (e.g. for True, Yes)
    - recalling that they are just symbols, not numerical values
  - By doing that, we can compute the (dis)similarity between two records described by binary categorical values building a **contingency table**
  - For example, how to measure (dis)similarity between the following two observations ?
    - $\mathbf{x}_1 = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0]$  and  $\mathbf{x}_2 = [0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0]$  (10 binary variables)

# Binary Variables

contingency table

		Data Object $\mathbf{x}_j$		
Data Object $\mathbf{x}_i$		0	1	Total
	0	$n_{00}$	$n_{01}$	$n_{00}+n_{01}$
	1	$n_{10}$	$n_{11}$	$n_{10}+n_{11}$
	Total	$n_{00}+n_{10}$	$n_{01}+n_{11}$	$n$

- **Contingency Table**

- In the central cells ( $n_{00}$ ,  $n_{11}$ ,  $n_{10}$ ,  $n_{01}$ ),  $n_{pq}$  is the no. of variables for which  $\mathbf{x}_i = p$  and  $\mathbf{x}_j = q$  (0 or 1)
  - For instance,  $n_{10}$  is the number of variables for which  $\mathbf{x}_i = 1$  and  $\mathbf{x}_j = 0$
- Therefore,  $n_{11}$  and  $n_{00}$  are the number of variables for which the observations agree, i.e., counts of agreements, while  $n_{01}$  and  $n_{10}$  are counts of disagreements
- Back to our example:  $\mathbf{x}_1 = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0]$  and  $\mathbf{x}_2 = [0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0]$  ( $n = 10$ )
  - These observations clearly agree on value "0" for variables 2, 3, 9 and 10, therefore  $n_{00} = 4$
  - Similarly,  $n_{11} = 1$  (variable 4),  $n_{01} = 2$  (variables 6 and 7), and  $n_{10} = 3$  (variables 1, 5 and 8)
  - Total agreement is  $n_{11} + n_{00} = 5$ , total disagreement is  $n_{10} + n_{01} = 5$

# Binary Variables

contingency table

		Data Object $\mathbf{x}_2$		
Data Object $\mathbf{x}_1$		0	1	Total
	0	$n_{00}=4$	$n_{01}=2$	$n_{00}+n_{01}$
	1	$n_{10}=3$	$n_{11}=1$	$n_{10}+n_{11}$
	Total	$n_{00}+n_{10}$	$n_{01}+n_{11}$	$n$

- **Simple Matching Coefficient (SMC):**

- It is the fraction of agreements in relation to the total number of variables ( $SMC \in [0,1]$ )

$$SMC(\mathbf{x}_i, \mathbf{x}_j) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}} = \frac{n_{11} + n_{00}}{n}$$

- In our previous example, we have  $SMC(\mathbf{x}_1, \mathbf{x}_2) = 5 / 10 = 0.5$

```
# SMC Coefficient in R (brute-force)
> x1 <- c("1","0","0","1","1","0","0","1","0","0")
> x2 <- c("0","0","0","1","0","1","1","0","0","0")
> n11 <- sum((x1 == "1") & (x2 == "1"))
> n00 <- sum((x1 == "0") & (x2 == "0"))
> n01 <- sum((x1 == "0") & (x2 == "1"))
> n10 <- sum((x1 == "1") & (x2 == "0"))
> SMC <- (n11 + n00)/(n11 + n00 + n10 + n01)
```

OR

```
# SMC Coefficient in R (using function table())
> x1 <- c("1","0","0","1","1","0","0","1","0","0")
> x2 <- c("0","0","0","1","0","1","1","0","0","0")
> CT <- table(x1,x2) # Contingency table
      x2
x1  0  1
 0  4  2
 1  3  1
> SSC <- (CT[1,1]+CT[2,2])/sum(CT)
```



# Simple Matching Coefficient (SMC)

$$SMC(\mathbf{x}_i, \mathbf{x}_j) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}} = \frac{n_{11} + n_{00}}{n}$$

- **Symmetric vs Asymmetric Variables:**

- SMC is appropriate for **symmetric** variables, since it counts both 1-1 and 0-0 matches equally
- For **asymmetric** variables, in which only non-zero values count, this is not appropriate
- For instance, let us suppose that our records  $\mathbf{x}_i$  and  $\mathbf{x}_j$  describe two patients and each of the variables describes the presence (1) or absence (0) of a symptom
  - Typically, patients are similar based on the symptoms that they have
  - Now consider the following patients:  $\mathbf{x}_1 = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$ ,  $\mathbf{x}_2 = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$ ,  $\mathbf{x}_3 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$
  - $\mathbf{x}_3$  does not have any symptom, yet it is as similar to  $\mathbf{x}_1$  as  $\mathbf{x}_1$  is to  $\mathbf{x}_2$  ( $SMC(\mathbf{x}_1, \mathbf{x}_3) = SMC(\mathbf{x}_1, \mathbf{x}_2) = 0.5$ ) !
    - **Exercise:** show it doing the calculations both manually as well as using R

# Binary Variables

contingency table

		Data Object $\mathbf{x}_j$		
Data Object $\mathbf{x}_i$		0	1	Total
	0	$n_{00}$	$n_{01}$	$n_{00}+n_{01}$
	1	$n_{10}$	$n_{11}$	$n_{10}+n_{11}$
	Total	$n_{00}+n_{10}$	$n_{01}+n_{11}$	$n$

- **Jaccard Coefficient:**

- An alternative to deal with asymmetric variables is the Jaccard coefficient ( $Jc \in [0,1]$ )

$$Jc(\mathbf{x}_i, \mathbf{x}_j) = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}$$

- This coefficient is obtained by removing the o-o matching counts (term  $n_{00}$ ) from the SMC
- By removing the term from both the numerator and denominator, the index keeps its range in  $[0,1]$
- In our previous example:  $Jc(\mathbf{x}_1, \mathbf{x}_2) = 1/6$  and  $Jc(\mathbf{x}_1, \mathbf{x}_3) = 0$ 
  - **Exercise:** show it both manually as well as using R (adapting the SMC code accordingly)!

# Exercises

$$\mathbf{X} = \begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- **Exercise 1:**

- Compute SMC and Jaccard for all pairs of observations in the dataset above (manually and in R)

- **Exercise 2:**

- Discuss what are the conditions required for SMC to reach its extreme values (0 and 1)
- Repeat for Jaccard

- **Exercise 3:**

- (a) Suppose that observations are students, and each binary variable indicates that a student is (1) or not (0) enrolled in a subject. There are n=20 variables (subjects). You want to assess similarity between students based on the subjects that they choose. Should you use SMC or Jaccard?
- (b) Repeat, now assuming that there are n=3 variables indicating the binary sex (M/F) and whether or not the student has a scholarship (Y/N) and identifies him/herself as indigenous (Y/N)

# Categorical (Non-Binary) Nominal Variables

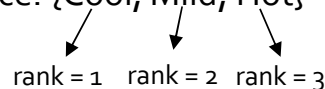
- Similarity measures such as SMC and Jaccard can be readily generalized to non-binary categorical variables. SMC is just the ratio between agreements and the total
- For instance, in the weather data,  $SMC(\mathbf{x}_1, \mathbf{x}_2) = 4/5$  since  $\mathbf{x}_1$  and  $\mathbf{x}_2$  agree in 4 out of 5 variables
  - Outlook, Temp, Humidity, and Play
- Now suppose that Play is an asymmetric variable for which only the value “Yes” is relevant
  - Any matching involving Play = “No” should be disregarded from both the numerator and denominator, which results in the Jaccard coefficient, given by  $Jc(\mathbf{x}_1, \mathbf{x}_2) = 3/4$

Weather Data from (Witten et al. 2016)

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

# Categorical Ordinal Variables

- For ordinal variables, two simple usual approaches are:
  - **Consider as Nominal:**
    - This essentially means that the information of order will be lost when computing distances
    - In the Weather dataset, for instance, by considering variable “Temp” as nominal when computing distances, a record with Temp = Hot is as different from a record with Temp = Mild than it is from a record with Temp = Cool, as far as that variable is concerned
  - **Convert to Numerical** (e.g., see `encode_ordinal()` from package `cleandata` in R):
    - The idea is to assign an increasing numerical value to the ranked ordinal values
    - General rule:  $\text{Numeric value} = (\text{rank} - 1) / (\text{no. of variable values} - 1)$ 
      - For instance: {Cool, Mild, Hot}  $\rightarrow$  Cool = 0, Mild = 0.5, Hot = 1

  
rank = 1   rank = 2   rank = 3

# Mixed Variables

- For datasets containing variables of different types (nominal, ordinal, numeric), two common approaches to compute (dis)similarity are:
  - **Convert different types into a single type**
    - The most common such a conversion is *categoric to numeric*
      - We've already preliminarily discussed *ordinal to numeric* conversion
      - We will discuss it more broadly (including *nominal to numeric* conversion) next
  - **Use a proximity measure designed for mixed-type data:**
    - For instance, **Gower's measure** and its variants
      - P.-N. Tan, M. Steinbach, V. Kumar, "Introduction to Data Mining", Addison Wesley, 2006
      - See also `daisy()` from package `cluster` in R

# Mixed Variables (Categorical to Numeric Conversion)

- Recall that when it is necessary to convert **ordinal** variables to numeric, one normally wants to keep the notion of order, otherwise there will be loss of information
- We previously saw that the easiest way to convert an ordinal variable to numeric is to replace each categorical value with a numerical value that respects the original order
  - For example, "Pain"  $\in \{0, 1, 2, 3\}$ , where 0 = None, 1 = Mild, 2 = Moderate, 3 = Severe
- Notice that the assignment of values is at the analyst discretion and must be chosen to reflect the desired interpretation for the application in hand

# Mixed Variables (Categorical to Numeric Conversion)

- For instance, the previous example implies that, from the perspective of any analysis tool handling var. "Pain", the difference between any 2 consecutive values is the same
  - e.g., the diff. between Mild and Moderate = the diff. between Moderate and Severe = 1
- In R, we can readily convert an ordered factor to numeric using **as.numeric()**
- For example:

```
> pain_levels = c("None", "Mild", "Moderate", "Severe")
> pain <- factor(levels = pain_levels, ordered = TRUE) # Declare an ordered factor that can take on 4 values (pain levels)
> pain[1:6] = c("Moderate", "Moderate", "Severe", "None", "Mild", "None") # Assign 6 different values to the variable as a vector
> pain
Moderate Moderate Severe   None      Mild      None
Levels: None < Mild < Moderate < Severe
> pain <- as.numeric(pain) # Convert to Numerical (None = 1, Mild = 2, Moderate = 3, Severe = 4)
> pain
3 3 4 1 2 1
```

- Once the var. is numeric, we can apply any desired transformation to change the default values 1, 2, ...

```
> (pain <- pain - 1) # Now None = 0, Mild = 1, Moderate = 2, Severe = 3
2 2 3 0 1 0
```



# Mixed Variables (Categorical to Numeric Conversion)

- The previous strategy would also "work" for non-ordered factors, i.e., nominal variables
- However, this is a classic pitfall for non-experienced analysts, which may inadvertently use this strategy unaware of the potential undesired consequences
- For example, consider a variable color, which can take values "Red", "Green" and "Blue"
  - Apart from certain particular physical interpretations of color, in most ordinary application domains Red is as different from Green as it is from Blue (e.g., for cars, bicycles, etc.)
  - If one applies the previous strategy, however, Red (1) will be twice as far from Blue (3) as it will be from Green (2), Blue will be 3 times larger than Red, yet 1.5 times larger than Green, etc.
  - If mathematical operations (e.g., distance computations) are performed with this variable, all these undesired relationships will be introduced into the analysis and may result in misleading conclusions

# Mixed Variables (Categorical to Numeric Conversion)

- Certain analysis tools can handle categorical data without the need for any conversion, and these may be preferred in the presence of categorical variables
- If conversion to numeric cannot be avoided, a commonly used **nominal to numeric** conversion technique is the **one-hot encoding** (so-called *1-of-n representation*)
- In this type of representation, each **value** of the original variable is transformed into a (asymmetric) **variable** on its own, which takes only two values, typically 0 and 1
- For example, variable  $\text{color} \in \{\text{Red}, \text{Green}, \text{Blue}\}$  is transformed into binary variables Red, Green, and Blue. A blue car takes values 0, 0, 1 for these variables, respectively. Similarly, a green car takes values 0, 1, 0, while a red car takes values 1, 0, 0
- We can easily create these new variables in R

# Mixed Variables (Categorical to Numeric Conversion)

- For example:

```
> mycolors = c("Red", "Green", "Blue")
> dataset <- data.frame(color = factor(c("Blue","Red","Red","Blue","Blue","Green"), levels = mycolors, ordered = FALSE))
> dataset
  color
1  Blue
2   Red
3   Red
4  Blue
5  Blue
6  Green
> # Conversion of nominal to numerical (one-hot encoding) using transmute() from package dplyr:
> dataset %>% transmute(Red = ifelse(color == "Red",1,0), Green = ifelse(color == "Green",1,0), Blue = ifelse(color == "Blue",1,0))
> dataset
   Red Green Blue
1    0     0    1
2    1     0    0
3    1     0    0
4    0     0    1
5    0     0    1
6    0     1    0
```

- **Exercise:** convert variable `Species` from dataset `iris` (base R package "datasets") to numeric

# Mixed Variables (Categorical to Numeric Conversion)

- **Note (encoding schemes / dense embeddings):**
  - Alternative encoding schemes for nominal to numeric conversion (other than “one-hot”) exist that require less than  $n$  variables (lessening the increase in dimensionality)
    - E.g., many functions in R automatically create **dummy variables** based on such encodings
      - the simplest of which is basically the same as *one-hot*, except that it drops 1 of the variables (since  $n-1$  variables suffice to uniquely discriminate between  $n$  nominal values, one of which is then represented as all zeros)
  - However, these encodings generally don’t ensure that *distances* between the encoded versions of each nominal value are the same, so they need to be used with caution
    - Such encoding schemes may have a major impact on distance-based DM algorithms (e.g., clustering)
    - This is the case of the modern, machine learning (ML) based **dense embeddings**

# Note on Categorical to Numeric Embeddings

- ML-based word embeddings such as `word2vec` and others can be used to produce dense, lower-dimensional embeddings of a categorical variable whose many values cannot be efficiently represented using one-hot encoding
  - See, e.g., the [TensorFlow Guide on Word Embeddings](#)
- The real-valued (non-binary) weights of the embedding can be **learnt** for the task in hand
  - Possibly alongside other coefficients of the model, such as in supervised learning tasks
    - E.g., word embedding layer in deep neural networks (DNNs)
- The weights can also have been **pre-trained** for the purpose of representation/transfer learning
- Despite their recognized effectiveness and widespread use in black-box machine learning approaches, the price to pay in data analysis/mining is the potential ***loss of interpretability***

# Similarity vs Dissimilarity

- Given a similarity (dissimilarity) within  $[0,1]$ , it is very easy to convert it into a dissimilarity (similarity) just by subtracting it from 1 ( $d = 1 - s$  or  $s = 1 - d$ )
  - The transformation is linear, it just flips the values “upside-down” without distorting them

# Similarity vs Dissimilarity

- When dissimilarity is given by a distance in an interval  $[0, d_{\max}]$ , transforming it into similarity demands first to rescale this interval to  $[0, 1]$  (as explained before)
- If  $d_{\max}$  is unknown or  $d_{\max} = \infty$ , however, the following options are available:
  - Using  $s = 1 - d$  will just flip the values up-down, but  $s$  will take negative values and  $\max = 0$
  - To keep  $s$  within  $[0, 1]$ , an alternative is  $s = 1/(1 + \alpha d)$  or  $s = e^{-\alpha d}$  (where  $\alpha > 0$ ), but this has the side effect that the values are non-linearly distorted and depend on a parameter ( $\alpha$ )

# Similarity vs Dissimilarity

	x1	x2	x3	x4
x1	0	4	4	6
x2	4	0	2	4
x3	4	2	0	2
x4	6	4	2	0

- **Exercise:**

- Given the **distance matrix** above, convert it into **similarity** within [0,1]

- **Note:** the following properties are desirable and are in general valid for similarity measures

- $s(p, q) = 1$  (max. value) if and only if  $p = q$
- $s(p, q) = s(q, p) \quad \forall p \in q$



# (Dis)similarity Measures

- In these notes we discussed broadly used *general-purpose* (dis)similarity measures that can be applied to a wide spectrum of application domains
- There are hundreds of other such (dis)similarity measures (M. M. Deza and E. Deza. Encyclopedia of Distances. Springer, 3rd edition, 2009)
- It is important to be aware that specialised measures also exist that may be more suitable in particular application scenarios. A few examples include e.g.:
  - Elastic measures for (long) time-series, such as [Dynamic Time Warping \(DTW\)](#) and variants
  - Measures for (short) gene-expression time-series in bioinformatics ([time-course measures](#))
  - Measures for networked data, such as [Structural Similarity](#) and variants
  - Measures for distributions (e.g. Wasserstein distance, KL divergence), and others...
- The choice of a suitable measure can be seen as *part of the KDD cycle (pre-processing stage)*

# (Dis)similarity Measures

*"The choice of the measure of dis(similarity) is important for applications, and the best choice is often achieved via a combination of experience, skill, knowledge and luck ..."*

Gan, G., Ma, C., Wu, J., **Data Clustering: Theory, Algorithms, and Applications**, SIAM Series on Statistics and Applied Probability, 2007

# Optional Reading

## **Section 2.4 "Measures of Similarity and Dissimilarity":**

P.-N. Tan, M. Steinbach, A. Karpatne, V. Kumar, "Introduction to Data Mining",  
Addison Wesley, 2020 (2<sup>nd</sup> Edition)

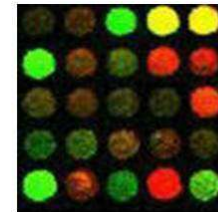
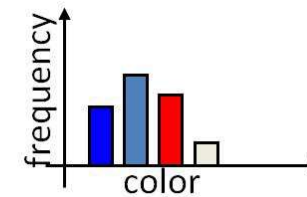
# Final Remarks on Data Representation

Adapted from:

[DM868 DM870 DS804](#)

(Arthur Zimek)

- ▶ image database  
↓  
image features
- ▶ gene database  
↓  
expression levels
- ▶ text database  
↓  
word encodings



Data	25
Mining	15
Feature	12
Object	7
...	

## Note that:

*Data mining methods work on the derived feature space no matter the original nature of the object – thus the mapping to a **meaningful** feature space is of paramount importance.*

## Final Remarks on Data Representation

Adapted from:

[DM868 DM870](#)  
[DS804](#)

(Arthur Zimek)

What a *meaningful space* is can be seen from different angles though...

E.g., while representation can arguably play a key role from a black-box ML perspective focused on **effectiveness**, such as predictive power (see figure)...

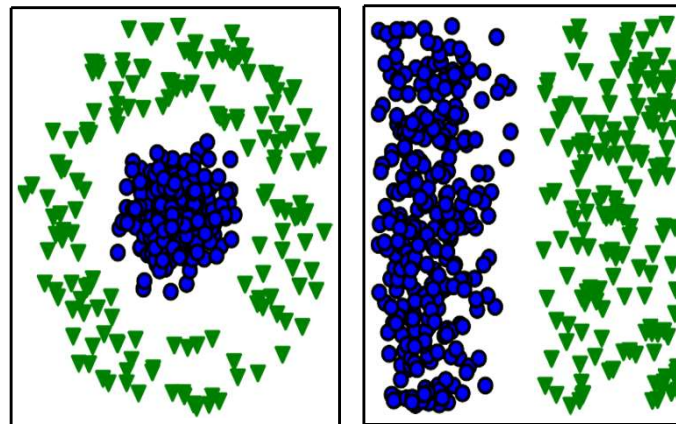


Figure adapted from Figure  
1.1, [Goodfellow et al. \[2016\]](#).

... there is usually a trade-off between effectiveness and **interpretability**, which is of paramount importance in many applications of data mining and analysis...

# References

1. P.-N. Tan, M. Steinbach, V. Kumar, "Introduction to Data Mining", Addison Wesley, 2006 (1<sup>st</sup> Edition), 2020 (2<sup>nd</sup> Edition).
2. Gan, G., Ma, C., Wu, J., "Data Clustering: Theory, Algorithms, and Applications", SIAM Series on Statistics and Applied Probability, 2007
3. I. Witten, E. Frank, M. Hall, C. Pal, "Data Mining: Practical Machine Learning Tools and Techniques", 4<sup>th</sup> Edition, Morgan Kaufmann, 2016
4. M. J. Zaki and W. Meira Jr., "Data Mining and Machine Learning: Fundamental Concepts and Algorithms", 2nd Edition, Cambridge Univ. Press, 2020
5. M. M. Deza and E. Deza, "Encyclopedia of Distances", Springer, 3rd Edition, 2009
6. I. Goodfellow, Y. Bengio, A. Courville, "Deep learning". MIT Press, 2016.