ADVANCED TECHNOLOGIES OF BIG DATA PROCESSING AND ANALYSIS

EDITOR

Adam DUDÁŠ

AUTHORS

Mihaela TİNCA UDRİSTİOİU
Adam DUDÁŠ
Alžbeta MİCHALÍKOVÁ
Fatih KILIC
Onder TUTSOY
Jarmila ŠKRINÁROVÁ
Silvia PUİU
Slaveya PETROVA

This material was funded by the European Commission within the Erasmus+ project
Applying some advanced technologies in teaching and research, in relation to air pollution
Project Code: 2021-1-RO01-KA220-HED-000030286

Funded by the European Union

University of Craiova

Paisii Hilendarski University of Plovdiv

Adana Alparslan Türkeş Science and Technology University

Matej Bel University, Banská Bystrica

ISBN
978-625-399-266-8

Book Title
Advanced Technologies of Big Data Processing and Analysis

Editor
Adam DUDÁŠ
ORCID iD: 0000-0001-5517-9464

Project manager
Mihaela Tinca UDRISTIOIU
ORCID iD: 0000-0002-5811-5930

Publishing Coordinator
Yasin DİLMEN

Page and Cover Design
Akademisyen Dizgi Ünitesi

Publisher Certificate Number
47518

Printing and Binding
Vadi Matbaacılık

Bisac Code
BUS070030

DOI
10.37609/akya.2633

Library ID Card
Udrıstıoıu, Mihaela Tinca and other.
Advanced Technologies of Big Data Processing and Analysis / Mihaela Tinca Udrıstıoıu, Alžbeta Mıchalikova, Fatih Kılıç. edıtor : Adam Dudas.
Ankara : Akademisyen Yayınevi Kitabevi, 2023.
170 page. : figure, table ; 195x275 mm.
Includes bibliography.
ISBN 9786253992668
1. Computer Technology -- Information Technology.

GENERAL DISTRIBUTION
Akademisyen Kitabevi A.Ş.
Halk Sokak 5 / A Yenişehir / Ankara
Tel: 0312 431 16 33
siparis@akademisyen.com

www.akademisyen.com

CONTENTS

INTRODUCTION

This handbook represents a result in the framework of the Erasmus+ project no. 2021-1-RO01-KA220-HED-000030286, titled "Applying some advanced technologies in teaching and research, in relation to air pollution." Four partners (Matej Bel University in Banská Bystrica, Slovakia, University of Craiova, Romania, Paisii Hilendarski University of Plovdiv, Bulgaria and Adana Science and Technology University in Adana, Turkey) have worked together to achieve this result. It aims to help STEM instructors to improve students' skills in working with data.

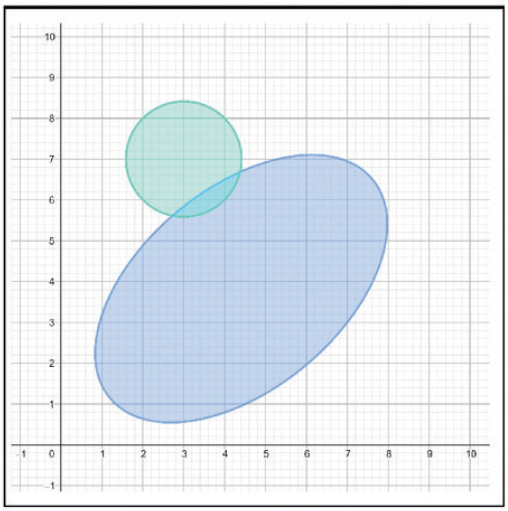
We are overwhelmed by the information that there is around us. Nowadays, it is necessary to know how to process data, extracting relevant information for each aim. At every second, computers, sensor networks, and satellites gather millions of values for physical amounts and parameters. Databases store and organize data and information, improving data quality. More than ever, information is power; from this point, STEM students must learn how to work with data. Companies require higher education to provide highly skilled graduates able to solve problems based on the information given by databases or using specialized programs or algorithms. In universities, STEM students should study how data sets are collected, analyzed, and interpreted. Also, they need to understand to make data classifications, approximations, and estimations. Finally, the Labor market asks STEM graduates to predict how processes evolve in space and time or make decisions. Machine Learning and Artificial Intelligence are standard terms in students' everyday vocabulary.

This handbook has ten sections, appendices, and references. The first part is about different types of data, their properties, data sampling methods, and how to process and analyze data. The following sections approach one of the most significant problems related to big data sets, data analysis. In analyzing big data, it is necessary to know how to use appropriate statistical analysis methods, data visualization, and other exploratory, predictive, and estimative methods. Different sections focus on approaches like machine learning, fuzzy inference, and neural network system. The appendix contains a description of the Iris dataset, examples of solutions to some problems, datasets on climate change or air pollution, and information about the impact of air pollution on human health. An example of a curriculum for a course in "Advanced technologies of big data processing and analysis" closes this handbook.

CHAPTER 1

DATA AND ITS PROPERTIES

This part of the handbook was written by Adam Dudáš from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.



	A	B	C	D
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5.0	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1
11	5.4	3.7	1.5	0.2
12	4.8	3.4	1.6	0.2
13	4.8	3.0	1.4	0.1
14	4.3	3.0	1.1	0.1
15	5.8	4.0	1.2	0.2
16	5.7	4.4	1.5	0.4
17	5.4	3.9	1.3	0.4
18	5.1	3.5	1.4	0.3

48° 44' 10.597" N 19° 8' 46.291" E

This handbook demonstrates simple data analysis methods using computer science techniques such as artificial intelligence, machine learning, or neural networks. In the following section of the work, we will deal with basic terms and concepts in data, its properties, processing, and analysis.

Data are technical, statistical, economic, or other types of messages or information that can be processed with the help of technical means. In our case, these technical means are computers. We approach data as objects that are integrated and shared within the system:

- **Data integration** – data can be placed in multiple files so that duplication is minimized and data from multiple files can be accessed simultaneously.
- **Data sharing** – each data object can be shared by multiple users (repeatedly and simultaneously).

The most important property of data is **persistence** - persistent data is data that exists even after the program is terminated. Whereas input data can be transformed into persistent, and output data can be transformed from persistent data, input data, or derived from it. The data derived from other data should not be persistent (we increase the costs related to the system's operation) - but sometimes, it is necessary.

It is essential to decide how the data in the records will be represented according to the specified types (while considering the most efficient storage possible). Most typical **data types** are:

- **Numerical data** - can be stored in various ways (binary, character, semi-logarithmic form, ...). It is often necessary to define the number of required bits/bytes for the number.
- **Strings** - can be stored in different character sets (ACII, UNICODE, EBDIC, ...).
- **Enumerators** - using character codes instead of strings (e.g., A instead of excellent, ...).
- **Units** - need to be adapted to the specific situation (nonsense: flight distance measured in millimeters).

These implementations of data types are not very interesting for us in the context of data analysis. In general, we will be talking about two types of data, distinguished by their contents:

- **Quantitative data consisting of numerical values** (height, distance, number, ...). Such data can be directly used in mathematical models, which is critical from the point of view of data analysis based on machine learning methods.
- **Categorical data consists of linguistic designations** of properties (gender, color, species, ...), which implies the need for specific data analysis methods. Some categorical data can be coded into quantitative, but such an operation is not always meaningful. An example of this coding could be something like *IF gender= male THEN gender = 1, IF gender = female THEN gender = 2* and so on. This makes sense in a way, but there are some questions, which discourage such coding:

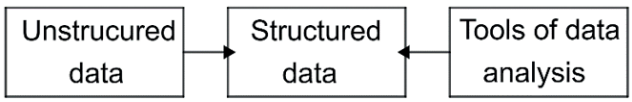
Since 2 - 1 = 1, is female - male = male?

What is the maximum value of gender?

The most important thing before analyzing any data is to have it structured properly. We distinguish **three types of data from the point of view of structure**:

- **Structured data** – data stored in tables or a file where it is possible to identify the same properties in the same order (columns) for each recorded object (row). The most frequently used structured data formats are csv, excel file, simple text file or SQL database.
- **Semi-structured data** - data is structured, but its shape is not fixed. We can therefore determine the same properties for each object (line), but these properties may not all be recorded for each record, or they may not be in the same order for all records (columns cannot be identified). This is typical for data from a set of sensors, mobile applications, and the like. The formats used for this data type are XML, JSON, or MongoDB.
- **Multi-structured/Unstructured data** – raw data of various formats. For example, raw sensor data, web logs, data from social networks, audio, video, images, 3D models, coordinates, and the like.

While working with unstructured data, we commonly use the following workflow:



Example: Conversion from unstructured data form to structured data (table). Let us have a collection of basic information on three students:

1. *Martin, male, 28.6.1983, year of study 2, 40 years*
2. *Jane, 1994-9-13., F, 1. year of st., 29 y.*
3. *Miriam, female, 5th of April 1992, second year of studies, age: 31*

The information is in an inconsistent form - the order of individual characteristics of people is different for each of the students, and the format of individual properties is also different. The data also contains easily computable data, which do not need to be stored (age). Therefore, when storing the data in a structured form (table) we need to unify the order and format of all properties (e.g., data in YYYY-MM-DD format).

Name	Date of birth	Year of study	Sex
Martin	*1983-6-28*	*2*	*M*
Jane	*1994-9-13*	*1*	*F*
Miriam	*1992-4-5*	*2*	*F*

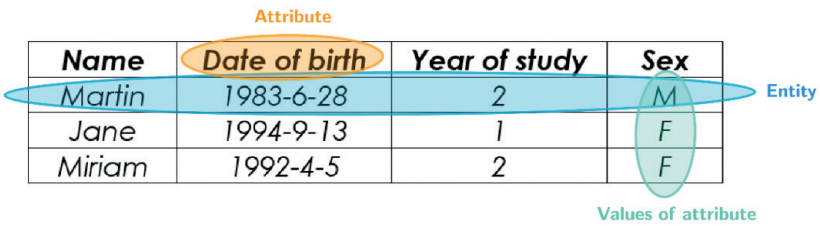
In the context of this handbook, we will be using various titles for the same objects in datasets. Therefore, we present a brief explanation of the terms below.

Entity, object, or record is a real-world object capable of independent existence and clearly distinct from other objects.

Attribute or property is a function assigning a value to an entity that determines some essential property of the entity (e.g., height, age, ...).

Dataset or table is a set of entities consisting of the same set of attributes.

The structuring of collected data is a basic method of data processing (see Section 2).



1.1 COUPLE OF WORDS ON THE TOPIC OF BIG DATA

In principle, it is always better to have a lot of data rather than a little (we can always throw some records out). We can call data big if their processing and analysis are not possible with conventional tools in a

practical time. Of course, the questions could be: *What is the practical time? What is a conventional tool?* Therefore, we use the definition of big data by enumerating its properties.

Data is commonly referred to as Big Data in the case it takes on properties referred to as 3V (the number of these "V"s increases over time, in some literature, 5V is the most basic model of view of big data):

- **Volume of the data** – The amount of data that can be gained from a few sources makes using simple relational database models unthinkable. We cannot represent the data with a simple table or set of tables and work on one machine. Therefore, the need to develop a more sophisticated computing infrastructure and implement optimized algorithms rises in importance. This system needs to implement high-performance, distributed, and cloud computing principles, non-relational databases which can store homogeneous and highly interconnected data, and artificial intelligence models for processing and analyzing the data.
- **Variety of the data** – Since the set of data which is considered in the case of real big data problems is not homogeneous, we need to be able to work with many file types and formats, e.g., simple text documents, audio files, video files, coordinates or computer models concerning more than two dimensions. Most of these data types are not storable in relational databases and demand high computing power and storage space to be successfully and conveniently stored and processed for further use.
- **Velocity of the data** – Big datasets are often live (or dynamic) datasets that change over time. This change of data over time occurs in all systems which implement so-called ambient data sources – sources that are always active and collect data, such as Internet of Thing sensors that measure the progression of the same set of values. This liveness of data and its processing, storage, and analysis create data streams that flow into the system. This brings one of the essential requirements for big data systems – the ability to collect, store, process, and analyze data in (almost) real-time.

On top of this problem of live data, we can identify issues by combining the dynamic and static parts of the big data, which causes several problematic events in the system.

Other than the volume, variety, and velocity of the data, the number of sources work with veracity and value:

- **Veracity of the data** – Since big data are often used in decision-making concerning the number of real-live entities, there is a severe need for trustworthy, reliable data. We need to consider metric, which measures our confidence in the data. This is important not only in the case of decision-making but also in the case of the reality of the data – generating big data is not hard and, therefore, can be easily used as a part of attacks that aim to overload the target system.
- **Value of the data** – as mentioned above, big data can be (and are) used to make decisions. The value of the data increases with the increase in the amount of the data related to the subject of study or the precision of estimation/prediction potential related to the data. This value can be monetary, business, human, research, or other significant.

These properties of big data bring forth several **problems related to their processing and analysis**.

The first of these problems is **the size of the data itself**. Their size is essential not only in the context of memory space, which is needed for storing the data itself, but also from the point of view of searching in the data and analyzing it. When working with such data, it is necessary to use high-performance, distributed, or cloud computing methods in conjunction with artificial intelligence algorithms from machine learning, fuzzy inference systems, and neural networks to gain knowledge from this data type.

In addition to the fact that these data are large, they are often composed of **heterogeneous partitions that may differ in several aspects** – the dimensionality of the data, the composition of the data, the

structure of the data, but also the measurements used. This inconsistency is a consequence of the fact that big data sets are often collected from several incompatible sources into one repository. Therefore, it is necessary for the data partitions to be reformatted (or, more precisely, for the individual data formats to be somewhat unified). This is represented by a sequence of simple tasks, which need to be executed on the data (such as conversion of measurements if needed) but also tasks that are more complex, e.g., identification of outliers and missing values. In the case of missing values, there might be action taken in order to compute the missing values – methods of machine learning and neural networks can be used for the estimation of the values or classification of the data.

The problem closely related to the heterogeneity of the big data sets lies in the **liveness of these sets of data**. In this case, we are measuring data with the use of ambient data sources (such as sensor networks), and these measurements are done on a high enough number of sensors in small enough time intervals; we are creating data streams that need to be processed and prepared for analysis in the system. Therefore, this system needs to be able to process and analyze datasets that change over time.

One of the most significant problems related to the big data sets is **analysis of the data**. The analysis needs to be supported by high-performance, distributed, or cloud computing, correct problem decomposition, and machine learning, fuzzy and neural network computing models. While analyzing the big data sets, we can use methods of statistical analysis, visualization of the data, and other methods of exploratory data analysis or predictive and estimative data analysis with the use of machine learning, fuzzy inference system of neural network approaches (see Section 2).

1.2 COMMON PROBLEMS IN DATASETS

Some common problems related to data were not described above - in the issues related to big data specifically.

As mentioned above, the amount of data constantly grows, so analyzing stored and processed data takes longer. However, the analysis is the essence of the data storage itself, and therefore it is impossible to avoid it. This brings with it the need for methods and procedures that allow us to acquire knowledge and support decision-making in the context of a large set of data.

Datasets intended for specific tasks are often created by combining data from several sources. These sources can be characterized by diversity in the formats and composition of individual data units. Thus we need a way to collect and unify such diverse data for further analysis. There is one more problem associated with this point. Since the data comes from different sources, a situation may arise when individual records will contradict each other (or not be consistent with each other).

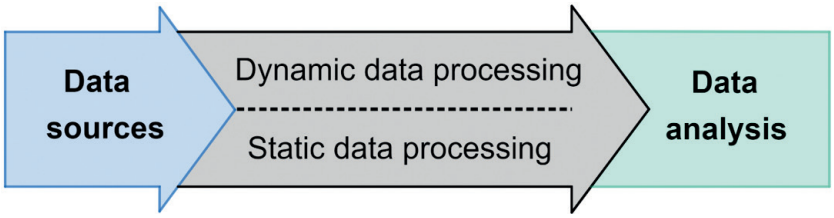
Data security is a big problem, even though it is not essential for us in the context of this handbook. Not every data source is secure and may not even be in accordance with the company's policy that wants to use it. In general, it is necessary to pay attention to the creation of authorization and authentication, monitoring of users who work with data, security of raw and acquired data, and protection of communication, i.e., data transfer.

Finally, problems particularly significant in creating predictions or estimations are missing data and outliers in data. Both problems are natural - in the case of missing values in data, there are situations when one of the required measured values in the dataset is missing. In the case of outliers, it is a natural situation where some measured values lie far outside the dataset's body. These two problems are the main content of Section 2 of this handbook.

CHAPTER 2

DATA PROCESSING AND ANALYSIS

This part of the handbook was written by Adam Dudáš from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.



When working with data, at least from the point of view of this handbook, we can identify two main activities - data processing and data analysis. Our main objective is to present information and examples of data analysis, which cannot be effectively performed without data that is suitable as an input to this process. Such preparation of data to a form-fitting for analysis without any problems is called data processing. The need for data processing and data analysis results from several characteristics of modern data:

- **Data sources** - nowadays, working with data sets created by combining smaller data sets collected from various sources is common. Data sets produced in this way can originate in different databases, sensors in one network, and a combination of these two approaches. Datasets that are composed of smaller parts bring with them very natural problems associated with this type of data structure:
  - homogenization of data structure,
  - working with missing values,
  - working with outliers.
- **Static data processing** – in modern systems, we perceive two types of data that can appear in the given system or which we might process in the system - the first type of data is static data. Static data is data that does not change over time. A common approach to processing such data is the so-called

batch processing. By default, these job batches include loading a file, processing the file, and writing the output to a new file without any manual interference by the user.

- **Dynamic data processing** – if the system uses ambient data sources (such as a constantly active sensor or a set of sensors), it must be able to capture and store this data in a time close to real-time. It doesn't matter what type of storage is used for it - it needs to support scaling due to the large volume of data. We also call such dynamically changing data sets data streams, which we must be able to process, filter, aggregate, and otherwise prepare for analysis. The data processed in this way is then sent for analysis. The problem of dynamic data processing is beyond the scope of this handbook, but it is quite an essential part of modern data systems.
- **Data analysis** - we perceive data analysis as the activity of acquiring knowledge for the needs of better decision-making in the context of a selected problem area, the possibility of predicting values based on collected data, or estimating unmeasured data. In the second part of this section, we describe the types of data analysis and the main problems of data analysis.

This handbook section is focused on data processing and selected problems regarding this action. The second part of the section focuses on an introduction to data analysis, which is then discussed in depth in subsequent sections of the presented handbook.

2.1 DATA PROCESSING

We do not always have the opportunity to work with a dataset ready for direct data analysis. Often (especially when it comes to our data), it is literally collected sets of information. Therefore, **cleaning** and **formatting** the data before analyzing it is necessary.

A note on this process - data processing and all steps described in this section of the text should always be performed on a copy of the original dataset, not on the dataset itself. We should also ideally use methods that are systematic and repeatable. After all, we do not want to lose our hard-earned data.

Internal consistency of the dataset

As mentioned at the beginning of this section of the handbook, the fact that modern datasets are created by combining several - smaller - datasets creates problems related to the internal consistency of the datasets themselves. This inconsistency can be perceived on two levels - the inconsistency of the data itself and the inconsistency of the dataset structure.

By default, several typical problems can cause data inconsistency:

- **Unit conversions** - when combining two datasets that use different units to measure attribute values (for example, centimeters and millimeters), it is necessary to unify the unit of measurement. It is also essential to unify datasets measured on continents that do not use the same measurement units - for example, to measure the same quantity, centimeters will be used in Europe and inches in the USA.
- **Numerical conversions** – numerical values recorded verbally need to be converted to numbers. This area of necessary conversions also includes typical problems specifying units within an attribute value.

- **Name conversions** – In recording the names of individuals, it is necessary to unify the way of recording names and surnames. The biggest problem in the case of datasets using name attributes from different continents is accented characters (e.g., *š, č, ä*).
- **Date and time conversions** – in the case of analyzes containing time information, it is necessary to unify the time recording format, especially the date in the given dataset.
- **Financial and currency conversions** – attribute values listed in different currencies must be unified to one of the currencies already present in the dataset.

The second case is the **inconsistency of the dataset structure**. The ideal method of storing data for further analysis is described in section 1 of this handbook - storing data as a table. However, this is not always possible to achieve simply - the problem in this case will be mainly missing values.

A dataset containing missing values is problematic to analyze using standard tools and software tools. Cells of the imaginary table where the value is missing are filled with *NULL* values that cannot be evaluated statistically and, at the same time, cannot be taken as values themselves (since $0 \neq NULL$). Therefore, it is necessary to deal with this type of problem in a certain way.

Missing and damaged data

For this handbook, data are viewed as measurements of real-world properties. These measurements are influenced by two factors: the data collection tool and the data processing method. In the case of both factors, a problem can arise, the consequence of which is **data loss or data damage**. If there is a problem with the data collection tool (burnt part of the sensor, lost records after a server outage, and so on), we talk about a data loss that cannot be reconstructed. The opposite is the loss or damage of data during their processing. If raw data is available, correcting the error is not a problem - we call this type of data loss or damage an **artifact**.

If our dataset is incomplete, it is necessary to identify the missing values and compensate them appropriately. The problem is that some of the missing values may not even exist. An example could be a value for an attribute that contains the time of arrival at a specified location in a situation where we have not yet arrived at that location.

Ways of working with missing values when the raw data are not available can be divided into a couple of types of compensation:

- **Substitution of missing value by other value** (0 / -1 / non-sense) – with such an approach, we would replace each missing value (NULL) with a selected, special value. This approach is not recommended - surrogate values can often be assumed to be correct and will be incorrectly interpreted in the analysis of the dataset. If, for example, we do not have a specified value for an employee's salary, we do not replace it with a value of 0 or -1, since the employee does not work for free or they do not pay for being able to come to work.
- **Dropping incomplete entities** – a slightly better case compared to the previous one could be the approach where we remove each incomplete record from the dataset. This approach is acceptable if we have enough data but it can still lead to biased results.
- **Computation of missing values (imputation)** – if we need to use records containing missing values, we can compute these values using one of the methods below. We also call this approach value imputation.
  - Imputation using the heuristic approach – if we know enough about the dataset and its relationships, we should be able to estimate the value of some attributes.

- Imputation by an average attribute value – this method replaces missing values with the average value for the given attribute. Using such a value is advantageous for several reasons, the most important of which is that average values of attributes are not strong in either direction and therefore have little impact on the predictive potential of the dataset. However, replacing the missing values with the average value of the given attribute is not always appropriate. This approach would be fine for an average salary, but an average arrival date at a given location does not make sense.
- Imputation by a random value of the attribute – for the missing value, we choose a random value of the given attribute that we have recorded in the dataset.
- Imputation using machine learning methods - the most sophisticated approach to the computation of missing data is using machine learning methods. However, these methods cannot be used with any dataset - or, more precisely, it is not possible to use them effectively for any dataset. Machine learning methods work based on correlations between individual values in the dataset. If these correlations are weak or non-existent, the estimates of the dataset values will be inaccurate. This approach is described in more detail starting in section 4 of this handbook.

Outliers

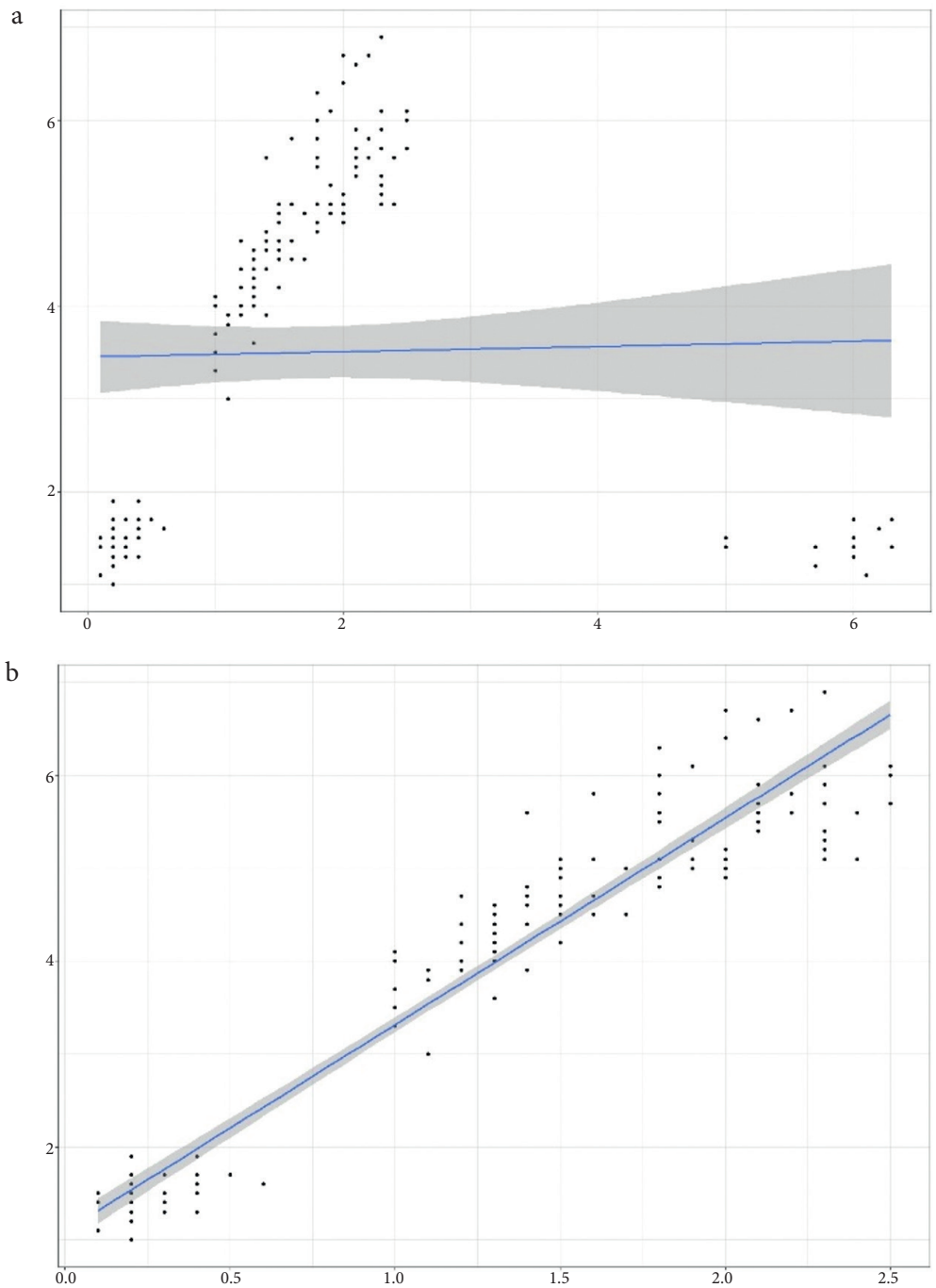
Outliers are values that lie outside the body of the dataset. In a normally distributed dataset, the probability of the occurrence of a value in the dataset decreases with the distance from the average value of the given dataset. However, the problem arises with datasets with a non-normal distribution. **Outliers occur in several ways**:

- a measurement error,
- typo during data processing,
- untrustworthy information that may further indicate the untrustworthiness of other records.

However, it is often a real value that deviates from standard situations (for example, periods of air pollution), so it is necessary to analyze the record as a whole.

The problem with outliers comes when trying to make any **generalizations** based on data that contains outliers. The figure below shows an attempt to describe the given dataset using a straight line. On the left side is a dataset containing 162 records, of which 12 are located significantly outside the body of the dataset. In this case, we see that the blue line that should pass through the center of the dataset completely misses it except for one point. On the right, we can see the same dataset after removing the given twelve outliers. The generalization result is much more satisfactory in this case.

If we want to make any generalization on the dataset, the outliers will act as a disturbing element, and therefore it is advised not to take such attribute values (and the records that contain them) into account, even if they are correct. As can be seen in the following figure - we want to describe the dataset using a line (linear function, in fact). In the case of the subfigure a, the line deviated due to the presence of outliers (lower right corner of the considered space). After removing these outliers, we can see a drastic increase in the accuracy of this generalization (subfigure b).



2.2 DATA ANALYSIS

Data analysis aims to **obtain useful knowledge from data** to support informed decision-making about the problem, prediction of events, and behavior of selected objects based on processed data. We recognize several types of data analysis, but within this handbook, we will only be interested in three basic, **most frequently used types**:

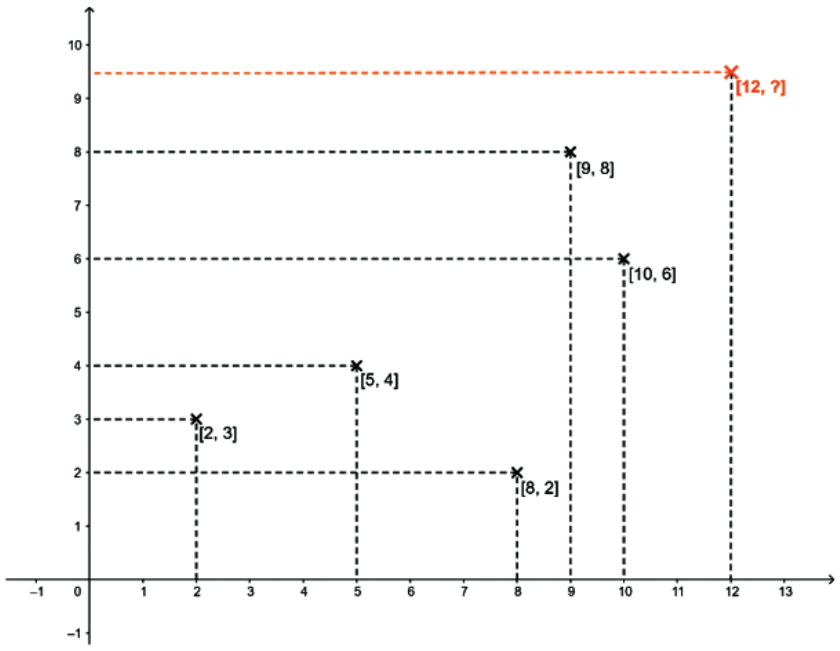
- **Descriptive and Diagnostic Data Analysis** - the simplest (and at the same time the most frequently used) method of dataset analysis. Descriptive analysis focuses on drawing conclusions (or gaining

knowledge) from data. It is most often used in the context of the description of the dataset and the measurement of the basic properties that the dataset describes - for example, the fulfillment of plans in the organization. Diagnostic Analysis aims to clarify why the events identified in the descriptive analysis occurred. Diagnostic analysis is often used because it creates connections between data and can be used to identify recurring patterns in the behavior of data objects. This type of data analysis is based on creating detailed information that can subsequently be used repeatedly when solving similar problems.

- **Exploratory Data Analysis** - the most natural type of analysis for humans is exploratory data analysis. It is focused on data analysis using exploration, most often with the help of data visualization. This analysis effectively identifies patterns and dependencies in data, but it is also important from the point of view of presenting the results of other analyses. In addition to the visual side of exploratory data analysis, we also include here actions associated with simplifying the dataset or representing the dataset - for example, dimensionality reduction, an operation where we project an n-dimensional data set into an m-dimensional data set while $m < n$.
- **Predictive Data Analysis** - predictive analysis is an extension of the aforementioned types of analysis. Its objective is to use the collected data to create logical predictions of event outcomes or predict and estimate values that we have not actually measured. In this type of data analysis, modeling methods based on statistics are used, which entails the need to use computing technologies to create prediction models. Note that the predictions that are the result of the models created during predictive analysis are only estimates for the given data set, and their accuracy, therefore, directly depends on the quality of the given data.

All these types of data analysis commonly work with just two basic types of problems to solve - the regression problem and the classification problem. The following part of this section is focused on describing these two problems.

Regression problem

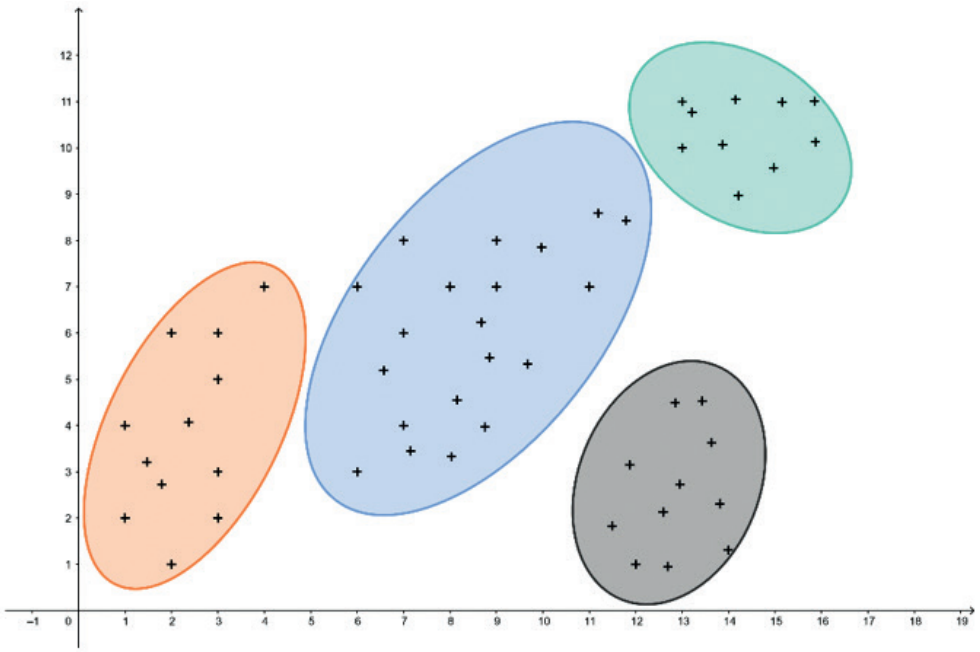


In the figure above, we can see a dataset that contains five points defined by the values of two attributes - these values are measured on the x-axis and y-axis, so we will refer to them as the values of the x and y attributes. This dataset then consists of points [2, 3], [5, 4], [8, 2], [9, 8], and [10, 6].

The regression problem, in this case, would be the task of estimating the real value of the attribute $y \in R$ if we know the value of x and the pattern from the previous (in this case) five points. Therefore, we have an entity containing a value for the attribute $x = 12$ and an unknown value for the attribute y, which we need to compute.

In general, we can define this type of problem as estimating or predicting the numerical value of the variable y based on the value of the variable x, where $x, y \in R$.

Classification problem



The general description of this problem can look like this: Given the pattern x and the space X, estimate which value of the associated attribute $y \in \{1,\ldots, n\}$ will be acquired by the pattern x.

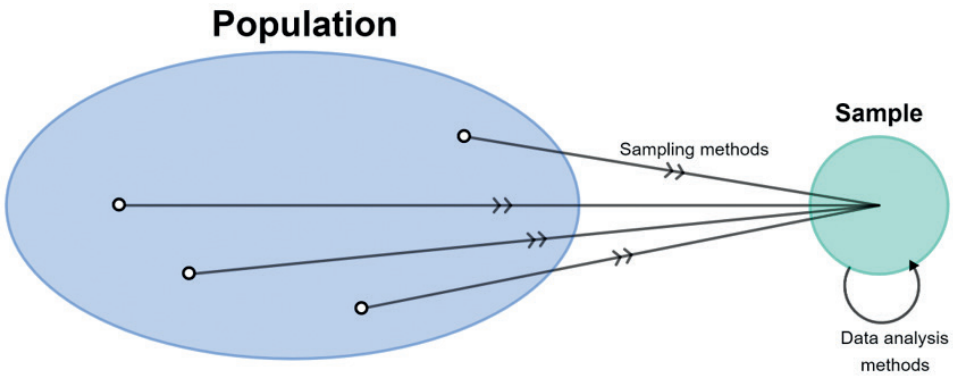
Such a description is slightly vague, it could be more understandable that in the problem of classification, we allocate entities to a set of pre-defined classes of individuals who are similar in a certain sense within one class. In general, we can identify three types of classification procedures:

- **hierarchical classification**, in which the classes are themselves classified, the process is repeated at various levels to form a tree,
- **partitioning**, in which the classes are mutually exclusive, thus forming a partition of the set of entities,
- **clumping**, in which the classes or clumps can overlap, and a clump and its complement are treated as different class types.

CHAPTER 3

DATA SAMPLING METHODS

This part of the handbook was written by Adam Dudáš from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia



Sampling can be defined as selecting a part of the population (dataset) that is the most representative of it so that it can be used for analysis and obtaining knowledge about the population. The technique used in sampling from the population is referred to as the sampling method.

Thus, a sample is defined as a part or **fraction of a population** selected from a population so that inferences can be made about the population, while the population is the totality of all subjects or subjects under study.

We know several well-known sampling methods. Most often, we divide them into **two groups** - probability and non-probability-based sampling methods. However, it is essential to say that the type to be used when selecting a sample depends entirely on the problem being solved. In general, however, we can say that:

- **non-probability methods** depend on the person who compiles the sample - so it is very easy to get results that a person would expect (even if they may not be true for the entire population).
- **probabilistic methods** more or less avoid this problem.

3.1 Non-probabilistic sampling methods

The selection of a sample from the population depends mainly on the judgment of the person who compiles the sample. Therefore, these methods can lead to distortions of some values compared to the population. Some methods of non-probabilistic sampling even depend only on the convenience of the person who compiles the sample - for example, the sampling method called Convenience sampling, where members of the population are selected based on the convenience of the compiler. Similarly, in a method called judgmental sampling, the sample is compiled based on the judgment of the compiler - for example, based on the non-data knowledge of the person compiling the sample.

A few non-probabilistic sampling methods are straightforward and are just a kind of "common sense" sampling. Therefore, in this handbook, we offer a more detailed description of only one of the non-probabilistic sampling methods.

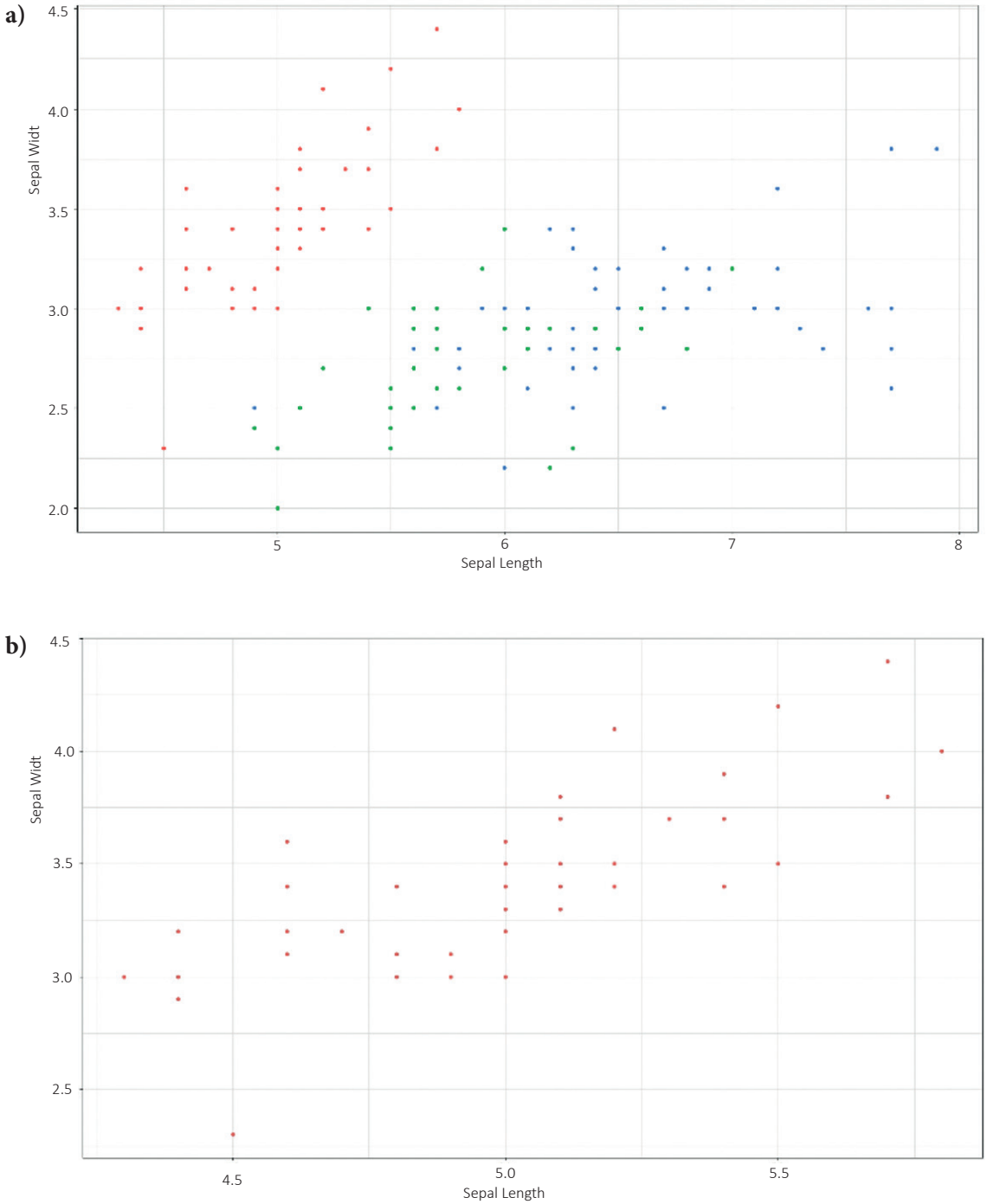
Purposive sampling method

In this sampling method, the person who draws the sample selects members of the population with a specific purpose for which the sample is drawn. Since all members of the population do not have the same chance of being included in the sample, we speak of a non-probabilistic sampling method.

An example of such sampling could be the need to compile an analysis of students in the third year of study in the computer science field of study, that is, to create a sample of third-year students of computer science students from the population of all students of all years in all fields of study. It is obvious that we will not want to include first, second, fourth- or fifth-year students in the sample. Likewise, we will not include in the sample students who study in fields such as applied mathematics, biology, or forensic chemistry.

To describe the results of individual sampling methods from this point in the chapter, we will use samples from the Iris dataset, described in Appendix A of this handbook. The task for the purposive sampling method could be as follows: *It is necessary to analyze the values of sepal length and sepal width for one specific type of flower - Iris Setosa.*

The figure represents a comparison between values of sepal length and sepal width in a) full iris dataset on the subfigure a (each class of the iris flower is marked with its own color) and b) a sample of the dataset consisting of one class - Iris setosa.



3.2 Probabilistic sampling methods

By this title, we refer to methods in which all members of the considered population have equal chances of being selected as part of the sample. These methods prevent (or reduce) the bias on the part of the sampler when adding objects to the sample, which was mentioned in the non-probabilistic methods section. Different types of probabilistic sampling methods are used in different situations to select sam-

ples from different populations.

These methods require the researcher to know the population under consideration, the appropriate sampling method, and how to use it in each situation encountered.

There are a few probabilistic sampling methods - multistage sampling, cluster sampling, systematic sampling, and so on. We will focus on four probabilistic sampling methods, which are simple and applicable to many solved problems.

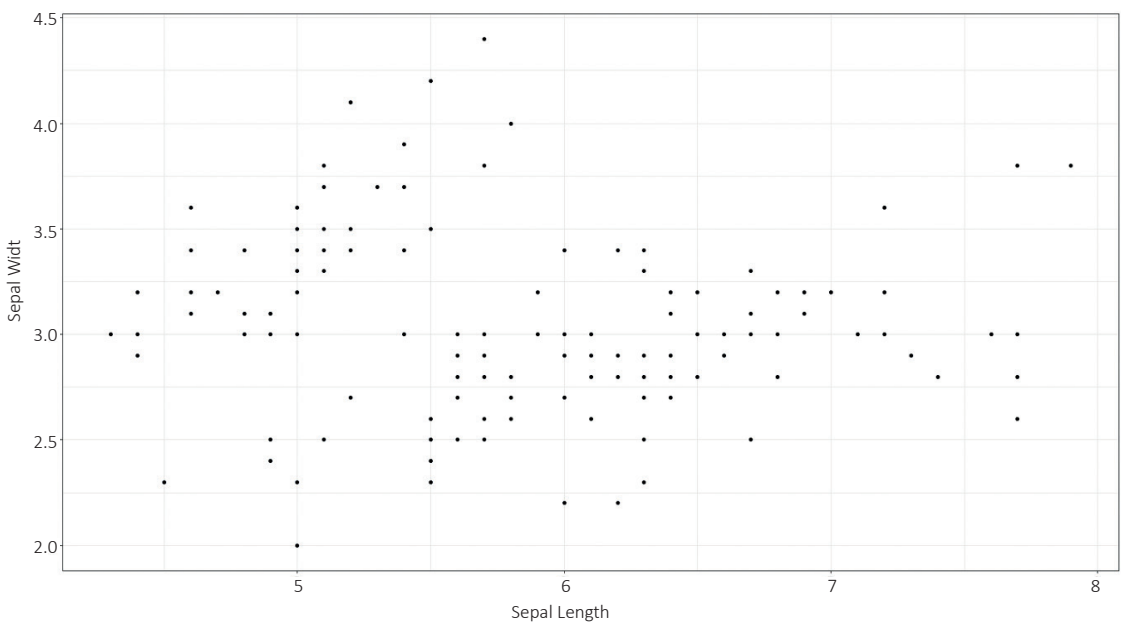
Simple random sampling method

This method is based on a random selection of individuals from the population. In other words, a certain sample is selected from any population without any mathematical model or logical decision-making. Since each individual (record) has the same chance to become part of the sample, this method is the most representative of the probabilistic sampling methods.

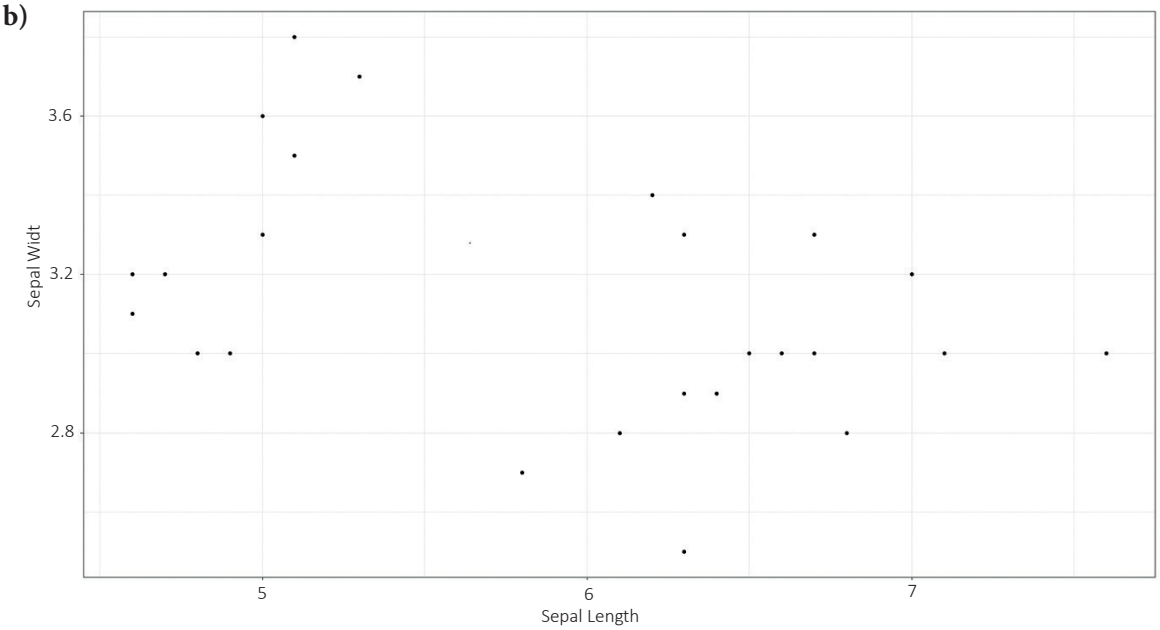
A simple random sampling method has only one input parameter - the desired size of the sample.

Example: Our population (subfigure a) contains 150 individuals (records), and we randomly select 25 representatives from it (subfigures b) - this set represents a simple random sample for us.

a)



b)



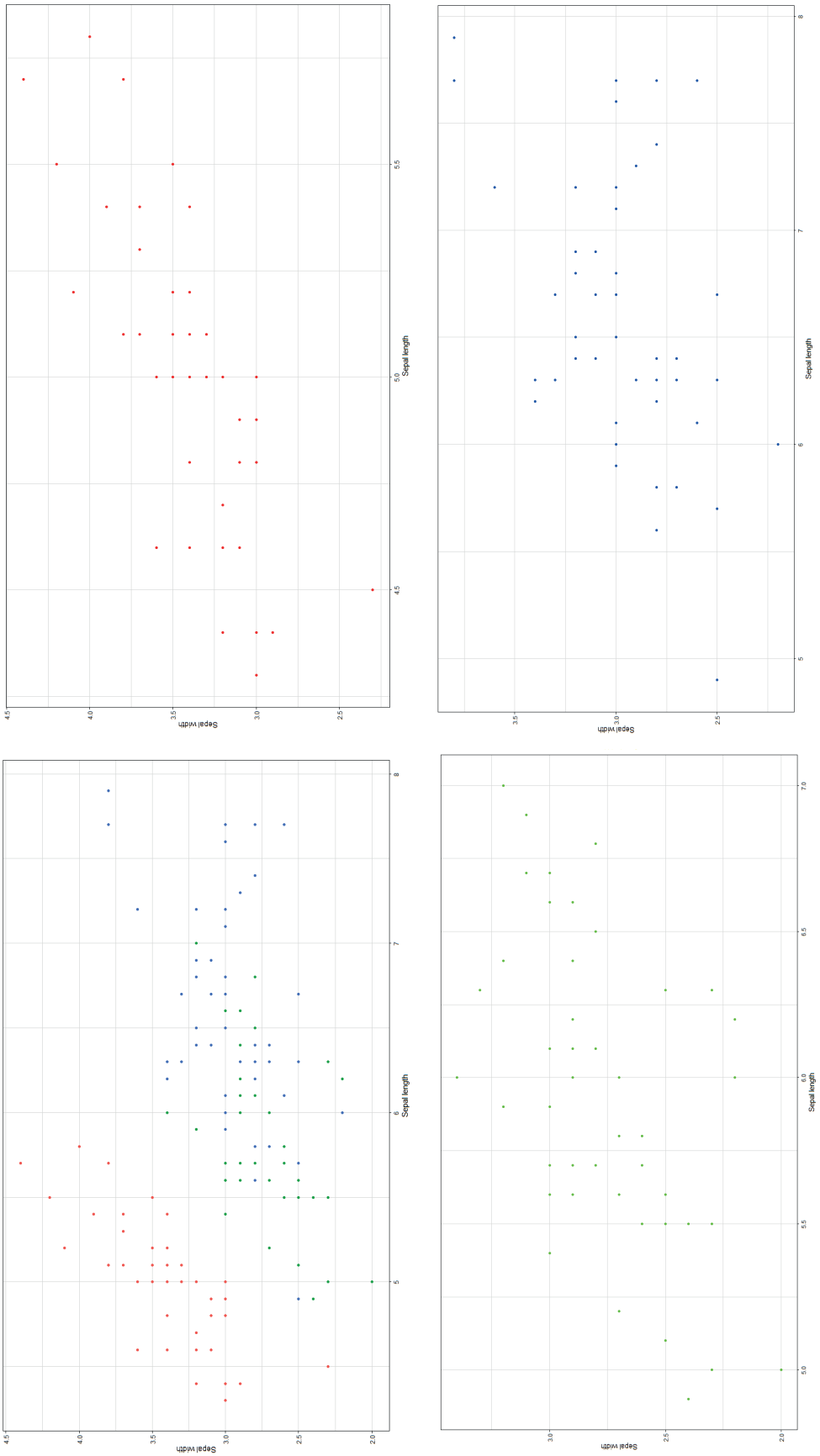
Cluster sampling method

A method in which an entire data set is divided into sections or clusters. Clusters are identified and included in the sample based on a certain attribute. This attribute is often categorical, such as hair color, gender, etc. The method is applicable to create samples suitable for analyzing already existing subsets in the data.

The cluster sampling method has only one input parameter - an attribute that should be used for data clustering.

Example: We divided our data set (top, left) according to the class attribute, which takes three values - iris setosa, iris versicolor, and iris virginica. Using the cluster sampling method, we can create three samples that can be used to analyze the characteristics of the individuals of the given classes. It is obvious that sample2 (top, rigth) is not suitable for drawing conclusions about the entire population, only about the subset of the population whose class attribute has the same value as sample2. An appropriate use of this method is, for example, the preparation of a statistical analysis describing individual clusters, which will allow comparison of the characteristics of iris flower classes.

Systematic sampling method


Sepal length
Sepal width
Sepal length
Sepal width
Sepal length
Sepal width
Sepal length
Sepal width

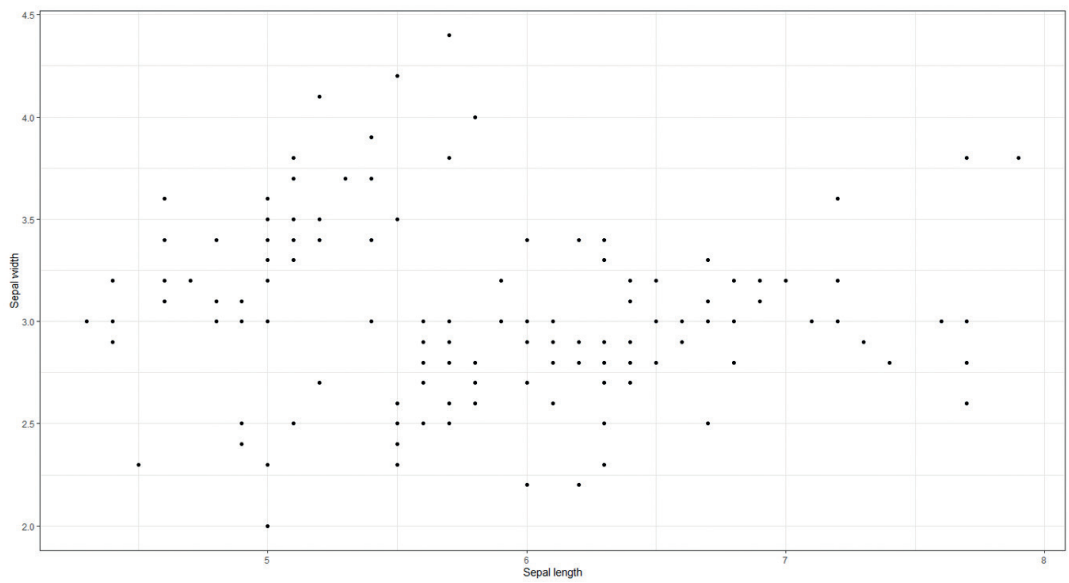
This method is used to select sample members from the population at regular intervals. This type of sampling method has a pre-defined scope and is, therefore, the least time-consuming sampling technique.

The systematic sampling method has two or three input parameters:

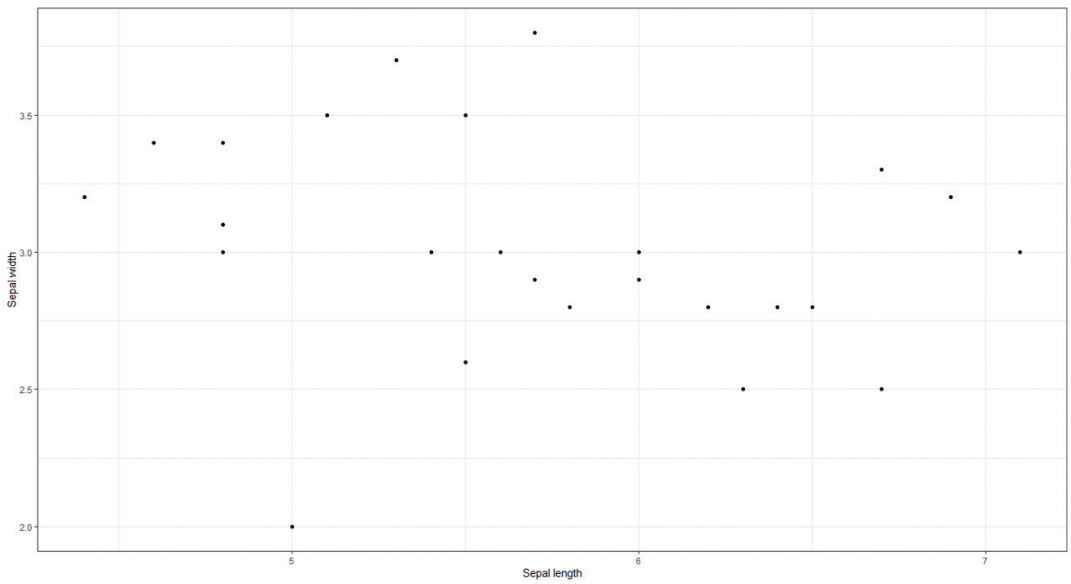
- selected starting point for creating the sample (the first individual that belongs to the sample),
- interval in which individuals are added to the sample, which creates implicit sample size,
- or interval in which individuals are added to the sample and the size of the sample we are creating.

Example: Since with the simple random sampling method, we selected 25 individuals, which represented the population we used; we also want to use the systematic sampling method to create a sample of 25 individuals. The original set consists of 150 representatives, and since 150/25 = 6 we will select every sixth individual (in case we start from the first record in the dataset). In the figure below, we can see the entire population (subfigure a) and the sample (subfigure b) consisting of 25 individuals selected by the procedure described above.

a)



b)



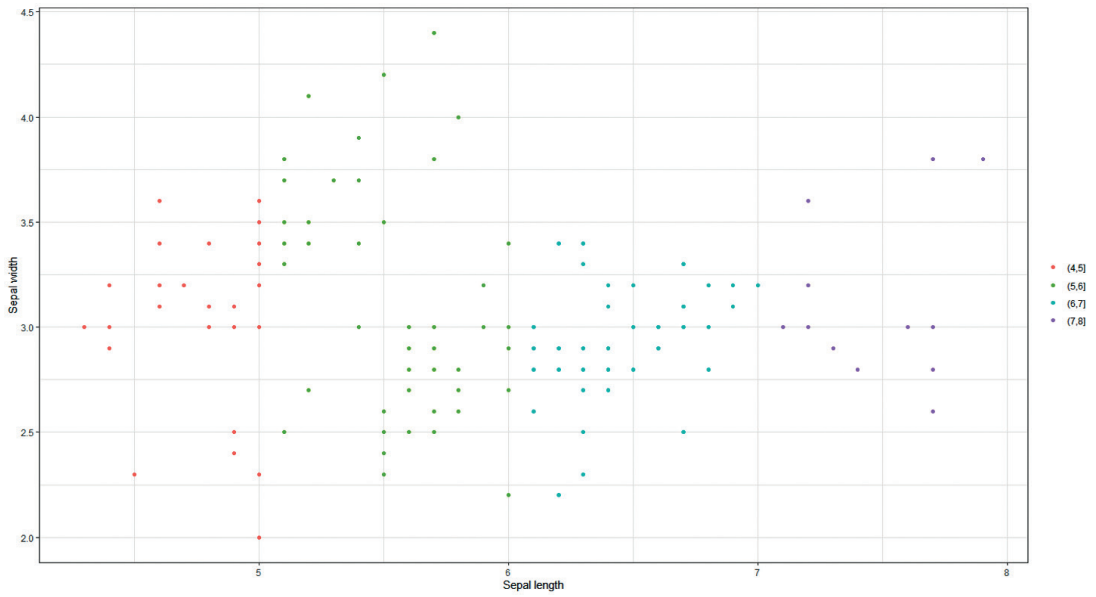
Stratified sampling method

With the stratified sampling method, the entire set of data is divided into smaller disjunct groups that represent the entire population. Compared to the cluster sampling method, this method creates groups in the data using a newly defined boundary on one of the attributes present in the original dataset. The cluster sampling method does not create these boundaries but uses one of the (categorical) attributes to identify groups in the data.

Example: The samples created using the stratified sampling method in the figure below can be defined as intervals defined on the Sepal length attribute. Each sample is different, but in each sample, there are representatives whose sepal length value is similar from the point of view of the selected method. In our case, we divided the samples by 1 cm from the smallest to the largest:

$$sepal\_length \in (4, 5]$$
$$sepal\_length \in (5, 6]$$
$$sepal\_length \in (6, 7]$$
$$sepal\_length \in (7, 8]$$

Thus, the figure contains four samples separated by color - sample1 marked in red, sample2 marked in green, and so on.



3.3 Couple of words on the topic of sample quality

Proper sampling is a necessary technique when working with big data (see section 1.1 for reference). The methods mentioned above create samples whose quality can be evaluated from several points of view. For this handbook, we present only two criteria to describe sample quality, and only one of them is truly critical for ordinary users (mainly meant as users outside of the computer science area):

- **Sample collection speed** - in today's world, we use modern software that often contains optimized functions and packages. Such functions include sampling methods, the implementation of which in the selected tool went through optimizations and methods that increased the effectiveness of the given function (this is almost guaranteed). If it is necessary to create a sample from a standard large data set (not big data-esque), the user will not come into contact with the problem of insufficient system performance, which would result in prolonged sample creation (or, in fatal cases, the inability to create a sample). However, when we work with truly big data sets, the standard system ceases to be efficient enough. From our own experience, we can give an example of creating a sample on a data set (population) with the size of one hundred million records, while each record contained sixteen attributes (note that this is not that big of a dataset). When using the systematic sampling method function with input parameter 4 (for creating a sample size of 25% of the population) in the R language on a standard user computer, we were unable to create a sample. This problem can be solved in several ways, the most common of which is the use of high-performance computing and cloud computing methods.
- **Representativeness of a sample** - an issue that is more important for the evaluation of sample quality than sample collection speed is the quality of its ability to describe the population from which it was created. As in the previous case, it is obvious that such a metric will not be universal - as was stated in the cluster sampling method description above, a sample of one cluster (one specific subset of data) is not suitable for drawing a conclusion about the entire population. In the case when it is appropriate to compare the characteristics of the sample and the population, we can proceed in several ways according to our objectives:

- **Statistical description of the sample** - in the case when we want to describe the data with a small number of values, we can compute critical statistical metrics. From these numerical values, we can derive knowledge suitable for further work with data.
- **Visualization of the sample** - big data are notorious for the complexity of their visualization. Therefore, it is a good idea to create a representative sample that contains fewer individuals and thus is easier to visualize.
- **Analysis of the predictive potential of the sample** - if our objective is to build prediction or estimation models based on machine learning, it is appropriate to analyze the predictive potential of individual attributes using methods such as correlation analysis or using decision tree models.

All of these approaches are described in more detail in section 4 of this handbook.

3.4 Couple of words on the topic of sample size

In the case we need to compile a sample from a given population, there could be a problem of **what size this sample should be in order to achieve the desired results** - to be able to accurately derive the knowledge we need. The answer to this question depends on the method used and on our objectives:

- In the case of sampling methods such as cluster sampling or stratified sampling, the answer is **given by the method used**. These methods create samples whose size is defined by the occurrence of a certain value in the data, and thus, in this case, it is not standard to consider a different sample size than the size of the cluster identified by the method.
- In other cases, especially with random sampling, it is necessary to use a **model that identifies the sample size suitable for our needs**. This model is defined by default for two types of populations - populations with a limited number of individuals or a population without a limit on the number of individuals. For our needs, we will consider the more natural of these versions - a limited population:

$$\bar{n} = \frac{\frac{z^2 \bar{p}(1-\bar{p})}{\varepsilon^2}}{1 + \frac{z^2 \bar{p}(1-\bar{p})}{\varepsilon^2 N}}$$

where

- $\bar{n}$ is the size of the sample,
- z is the so-called z-score which reflects confidence level, most commonly set to 90%, 95%, or 99% with z-score coefficients as presented in the following table:

Confidence level	z-score
90%	1.65
95%	1.96
99%	2.58

This table is a typical example of z-score tables that contain pre-calculated z-score values and can be searched online for other confidence level values.

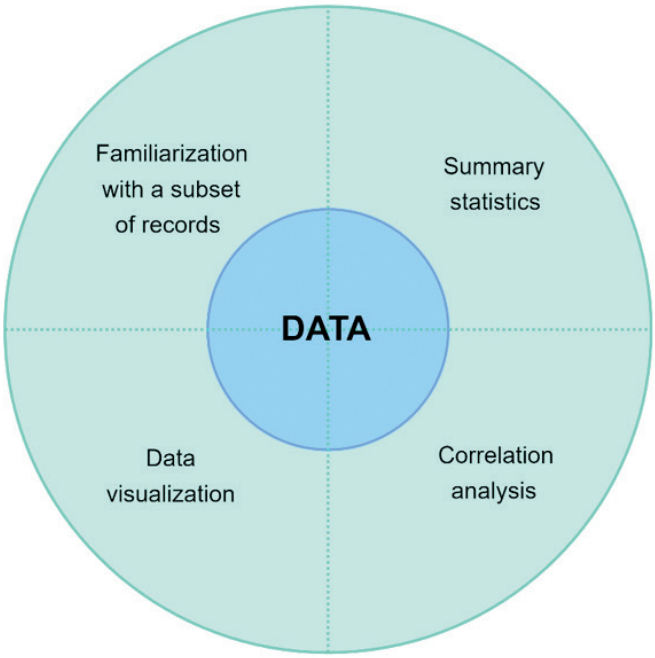
- $\overline{n}$ is the proportion of population - a percentage (or fraction) of the population associated with the researched problem (standard value for an unknown population is set to $p = 0.5$).
- ε is the margin of error set by the user.
- N is the size of the population used.

The easiest way to calculate the sample size is to use freely available online sample size calculators, which work on the principles listed above.

CHAPTER 4

BASICS OF EXPLORATORY DATA ANALYSIS

This part of the handbook was written by Adam Dudáš from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.



As mentioned in the previous sections of this handbook, in the process of data analysis, we can work with the entire dataset or with a sample created using the methods mentioned in section 3 of this text. For basic, descriptive data analysis, we use descriptive statistics methods that provide tools to capture the characteristics of a set or sample of data. Descriptive statistics methods are based on aggregation methods for representing subsets of data, for example, attribute average value, minimum, frequency, or sum of values. Such methods can be called aggregation as data reduction methods.

When analyzing data with the use of descriptive statistics methods, we mainly use the following three concepts:

- **Central tendency measures**, with which we look for data centers around which data are clustered or distributed.
- **Variability measures** describe the distribution of data in considered space, i.e., how far individual measurements are from the center identified using central tendency measures.
- **Correlation analysis** is based on the computation of coefficients that describe the prediction potential between individual attributes in the dataset. These coefficients are essential in the creation of machine learning models but also in data visualization, which is an essential part of this section of the handbook.

In this section of the handbook, we will confront the first version of data analysis, which is very natural from a human point of view - **Exploratory Data Analysis** (EDA). As the name suggests, it is data analysis using an exploration of data to find patterns and trends in a given population or sample. In its most basic form, this type of analysis is performed by visual exploration, and therefore data visualization methods will be an important part of such analysis.

In order to know which parts of our dataset are appropriate to visualize and which are not, we use the basic knowledge acquired through the methods of descriptive statistics.

4.1 Basic statistic methods

From the point of view of basic statistical methods, we can identify methods with which we measure centrality in data - we look for data centers around which data are clustered or densely distributed. The most common of these methods are:

- **Average or mean** is the average value of array elements. The average is suitable for characterizing symmetrically distributed data without outliers (e.g., height, weight). Symmetrically distributed data are those where the number of elements of the dataset should be similar below and above the mean limit, ideally the same. The relation for computing of average value is as follows:

$$\mu_A = \frac{\sum_{i=1}^{n} Ai}{n},$$

where μ_A is the average value of attribute A, n is the number of entities containing the attribute A, and A_i is i-th value of this attribute.

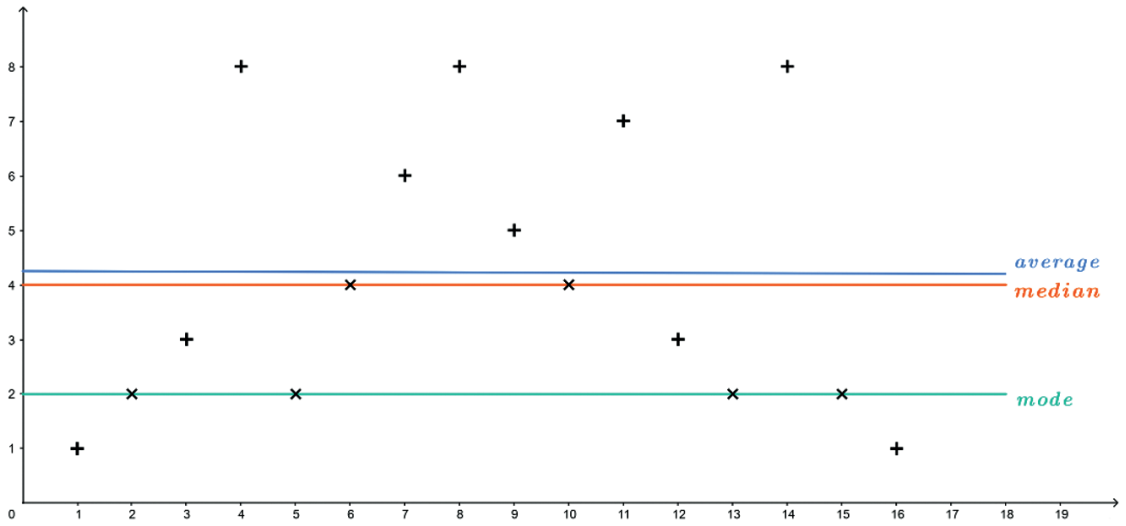
- **Median** is the middle value of the sorted array. The median is extremely symmetric, meaning that there is the same number of elements below the median as above. One exception to this rule is the datasets containing an even number of elements (then we choose one of the two middle values as the median - in a reasonable dataset, they should be quite close to each other). Unlike the average, the median is a real value of an attribute, so it is more suitable if the data contains outliers and is asymmetrically distributed (for example, the wages of employees in a particular area). The median is computed with the use of the following relation:

$$median_A = \frac{(n+1)}{2} \; th \; element \; of \; a \; sorted \; set \; A,$$

where n is the number of entities containing the attribute A.

- **Mode** is the most frequently occurring element in the attribute. However, this measure is difficult to use and inaccurate in most analytical tasks. An example could be the mode for the wages mentioned, which would mostly equal 0 since exactly 0 is earned by most people - unemployed, children, and pensioners.

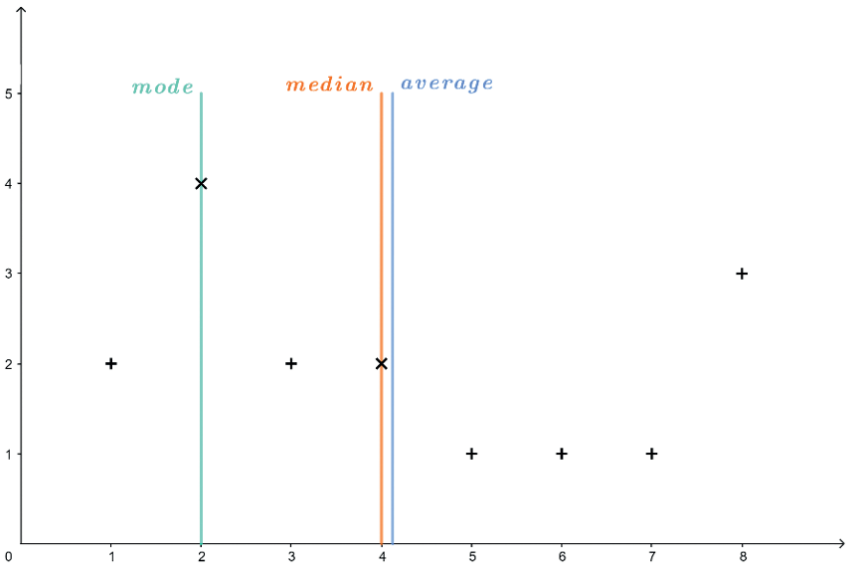
In the following figure, we offer a visualization of centrality measures for a simple dataset. We can see a fairly typical behavior of average and median values in data that are normally distributed in the considered space - that is, these values are quite close to each. Mode value is hard-to-predict since it is the most frequently occurring element value of the attribute (therefore, it can be high, it can be low, it can be somewhere in the middle).



In addition to these standard measures, the **frequency of values** in an attribute is highly important. By the title of **frequency distribution**, we mean a list, table, or graph that shows the frequency of occurrence of various outputs in a sample (dataset). Each entry in the table contains the frequency (or number) of value occurrences in a given group or interval. An example of such a frequency distribution for the simple dataset used above looks as follows:

value	frequency
1	2
2	4
3	2
4	2
5	1
6	1
7	1
8	3

This table can be mapped to the **graph** below. Such visualization of frequency graphs is essential, especially from the point of view of getting to know the data and possible detection of outliers.



The other side of the coin with standard statistical measures is data variability in considered space. The most common measure of variability is the so-called **standard deviation** (σ), which is defined as the sum of the squared differences between the individual elements of the attribute and is the average value:

$$\sigma_A = \frac{\sqrt{\sum_{i=1}^{n}(A_i - \mu_A)^2}}{n-1},$$

where μ_A is the average value of attribute A, n is the number of entities containing the attribute A, and A_i is i-th value of this attribute.

MeasurΣe similar to the standard deviation is **variance** computed as:

$$V = \sigma^2$$

Example: Let us have the following simple dataset consisting of one attribute with five measurements A = [20, 60, 40, 70, 50]. Let's calculate the mean, median, and standard deviation for this dataset.

$$\mu_A = \frac{\sum_{i=1}^{n} A_i}{n} = \frac{20+60+40+70+50}{5} = \frac{240}{5} = 48$$

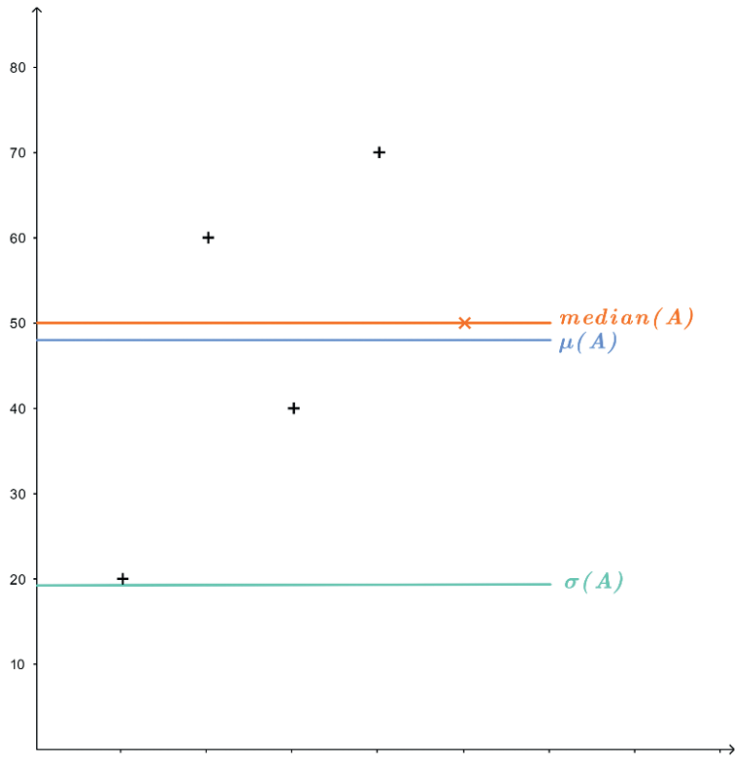
$$\text{median}_A = \frac{(n+1)}{2} = \frac{6}{2} = 3rd \quad \textit{lement of sorted set} \rightarrow [20, 40, 50, 60, 70] = 50$$

$$\sigma_A = \frac{\sqrt{\sum_{i=1}^{n} (A_i - \mu_A)^2}}{n-1}$$

$$= \frac{\sqrt{(20-48)^2 + (60-48)^2 + (40-48)^2 + (70-48)^2 + (50-48)^2}}{4} = \frac{\sqrt{1480}}{4}$$

$$\sqrt{370} \approx 19.235$$

This standard deviation is comparatively high, which is natural since the dataset itself is very scattered. The visualization of these measurements is in the figure below.



Centrality and variability computation methods serve to describe the dataset using a small number of values. Such an approach to dataset description can also be called **description by aggregation**.

Example: Let us have attribute A = [1, 2, 3, 8, 2, 4, 6, 8, 5, 4, 7, 3, 2, 8, 2, 1] (presented at the beginning of section 4.1). We can describe this dataset with the use of three aggregated values, e.g. (min(A), μ(A), max(A)), therefore A = (1, 4.125, 8).

The last indicator from simple statistical metrics is the **distribution of dataset data in space**. The use of the average value of the attribute and the standard deviation characterizes this metric. In a normally distributed dataset, at least $(1 - 1/k^2)$*-th* of the points lie at a distance of $k\sigma$ or less from the average. Such a dataset does not contain outliers and is an excellent candidate for machine learning and artificial intelligence data analysis methods.

Example: For our familiar attribute A = [20, 60, 40, 70, 50], we can compute the distribution as follows:

$\mu A = 48$

$\sigma A \approx 19.235$

$2\sigma = 2 * 19.235 \approx 38.47$

We can see that at least 3 of the values should be at most 38.47 units away from the average value (48). This is true for all measurements of attribute A.

4.2 Correlation analysis

The basic statistical values described in the previous section are important indicators to describe the given data. However, from the point of view of data analysis objectives, the so-called correlation analysis is a much stronger metric.

In the case our dataset contains more than one numerical attribute, we can measure the correlation between the two-element subsets of this dataset. Let us have two attributes of dataset A – A_1, A_2. These attributes correlate with each other when attribute A1 has **predictive potential** for attribute A_2. Such predictive potential speaks of the **presence of trends and patterns** in the dataset and the possibility of building analytical models that work with the data.

We measure the correlation of two variables using the **correlation coefficient** $r(A_1, A_2)$, which presents how much attribute A_1 is a function of attribute A_2 and vice versa. This correlation coefficient can take values from the interval [−1, 1] while:

- **1** indicates the **complete correlation** of two attributes. In other words, when the value of attribute A_1 increases, the value of attribute A_2 also increases. If there is a complete correlation between the values of two variables, we are talking about a strong predictive potential, and thus these attributes are suitable for mutual prediction.
- **0** indicates the worst situation from the point of view of the correlation of two values, which we refer to as **non-correlation**. When the correlation coefficient between two attributes is close to or equal to 0, these are independent values that are unusable from the point of view of building analytical models.
- **-1** is the opposite of complete correlation, which we call **anticorrelation**. In this case, we can identify a trend in which, as the value of attribute A_1 increases, the value of attribute A_2 decreases, or vice versa. As in the case of complete correlation, this is a satisfactory condition for building analytical models.

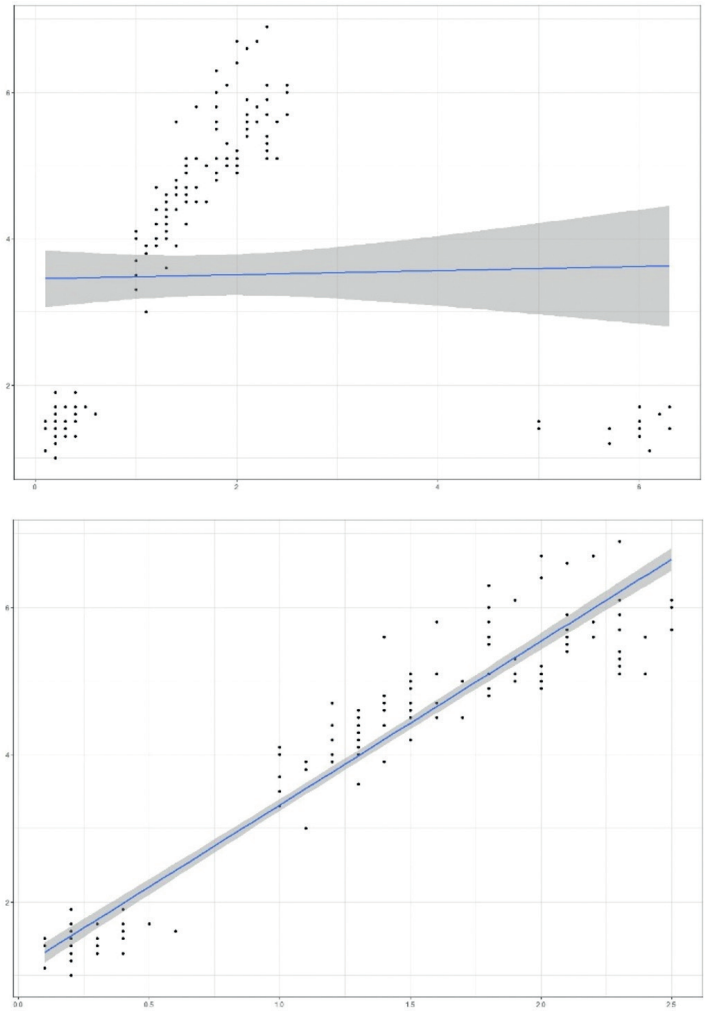
We use two standard methods to analyze correlations and measure correlation coefficients - of course, there are many more such methods. We will focus on the Pearson correlation coefficient and Spearman rank correlation coefficient for our purposes.

Pearson correlation coefficient

The first and strongest coefficient used to measure the correlation between two dataset attributes is the Pearson correlation coefficient. This coefficient is focused on the **linear prediction of values** and the relationship between attributes A and B. It is described using the following relation:

$$r = \frac{\sum_{i=1}^{n}(A_i - \mu(A))(B_i - \mu(B))}{\sqrt{\sum_{i=1}^{n}(A_i - \mu(A))^2}\sqrt{\sum_{i=0}^{n}(B_i - \mu(B))^2}},$$

where $\mu(A)$ is the average of attribute A, similarly $\mu(B)$ is the average value of attribute B, and n is the number of measurements (vertical size of the dataset). This apparent dependence on the average value brings the most significant disadvantage of the Pearson correlation coefficient - sensitivity to outliers (as shown in the figure below).



Using the Pearson correlation coefficient, we are looking for a line that describes the values of the given attributes. In the left image, we can see a visualization comparing the values of two attributes from one dataset, in which there are outliers (bottom, right). We can also see that the line that we have fitted the data with misses it completely and significantly (except one data point) - from this, we can conclude that the Pearson correlation coefficient is not suitable for measuring the predictive potential in such a dataset. On the right side, we present the same attributes of the dataset after removing outliers. We can see that, in this case, the line is descriptive of the trends present in the data.

Therefore, the **Pearson correlation coefficient can be used when** attributes *A* and *B* contain:

- linear relationships,
- normal (Gaussian) distribution,
- no outliers.

Spearman rank correlation coefficient

To deal with **datasets that contain nonlinear relationships** with outliers, we can use a different type of correlation coefficient - specifically, the Spearman rank correlation coefficient. This method of measuring the correlation between attributes creates a **hierarchy** (ranking) of individual attribute values for its functionality.

Example: Let us have attribute A = [a0 = 4, a1 = 8, a2 = 2, a3 = 6]. The abovementioned hierarchy or ranking would look something like this:

since a1 > a3 > a0 > a2 the rank(a1) = 1, rank(a2) = 4, and so on

In this way, we measure the **monotonicity of the values** within the attribute. Therefore, the Spearman rank correlation coefficient is most suitable for datasets with monotonic relationships between attributes - if one attribute value increases, the other never decreases or vice versa. On the other hand, this type of correlation coefficient is not recommended to be used if there are repeated values (meaning the same rank) in the dataset. This effect is attenuated with increasing dataset size.

Spearman rank correlation coefficient is computed as:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)},$$

where $d = rank(a_i) - rank(b_i)$ and n is number of entities in considered attributes.

Example: Warning – the following example is popular amongst students; they might remember the example more than the correlation ideas themselves. Let us have two attributes - the price of the kebab and the distance of the kebab place from the university.

Index	*Distance in meters*	*Price in euro*
1	*10*	*4*
2	*70*	*3.50*
3	*85*	*3.30*
4	*100*	*3.20*
5	*130*	*3.80*
6	*195*	*2.90*
7	*215*	*3.10*
8	*300*	*3.90*
9	*420*	*3.15*
10	*505*	*3*

First, we compute the value of the Spearman rank correlation coefficient. This method requires the creation of a ranking for both attributes and the computation of the values of d and d^2 as follows:

Index	*Distance in meters*	*Rank(distance)*	*Price in euro*	*Rand(price)*	*d*	d^2
1	*10*	*10*	*4*	*1*	*9*	*81*
2	*70*	*9*	*3.50*	*4*	*5*	*25*
3	*85*	*8*	*3.30*	*5*	*3*	*9*
4	*100*	*7*	*3.20*	*6*	*1*	*1*
5	*130*	*6*	*3.80*	*3*	*3*	*9*
6	*195*	*5*	*2.90*	*10*	*-5*	*25*
7	*215*	*4*	*3.10*	*8*	*-4*	*16*
8	*300*	*3*	*3.90*	*2*	*1*	*1*
9	*420*	*2*	*3.15*	*7*	*-5*	*25*
10	*505*	*1*	*3*	*9*	*-8*	*64*

The values from this table can therefore be added to the relationship for computation of the Spearman rank correlation coefficient:

$$\sum d^2 = 256$$

$$n = 10$$

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} = 1 - \frac{6 * 256}{10(100-1)} = 1 - \frac{1536}{990} = 1 - 1.55 = -0.55$$

So, we found out that there is a correlation of -0.55 between these two attributes, which could be called a moderately strong anti-correlation.

Let us also compute the value of the Pearson correlation coefficient. To calculate this type of coefficient, we need to determine the average value of the distance μ(distance) and the average value of the kebab price μ(price):

- *μ(distance) = 203, hereinafter referred to as μ(d)*
- *μ(price) = 3.39, hereinafter referred to as μ(p)*

Our table will contain more columns, but all of these are only precomputations of parts needed in the final Pearson correlation coefficient relation[1]*:*

index	d	p	d − μ(d)	p − μ(p)	$(d - \mu(d))^2$	$(p - \mu(p))^2$	(d − μ(d)) (p − μ(p))
1	10	4	-193	0.61	37 249	0.3721	-117.73
2	70	3.50	-133	0.11	17 689	0.0121	-14.63
3	85	3.30	-118	-0.09	13 924	0.0081	10.62
4	100	3.20	-103	-0.19	10 609	0.0361	19.57
5	130	3.80	-73	0.41	5 329	0.1681	-29.93
6	195	2.90	-8	-0.49	64	0.2401	3.92
7	215	3.10	12	-0.29	144	0.0841	-3.48
8	300	3.90	97	0.51	9 409	0.2601	49.47
9	420	3.15	217	-0.24	47 089	0.0576	-52.08
10	505	3	302	-0.39	91 204	0.1521	-117.78

So, we can compute the Pearson correlation coefficient as follows:

$$\sum((\mathrm{d} - \mu(\mathrm{d}))^2) = 232\,710$$

$$\sum((\mathrm{p} - \mu(\mathrm{p}))^2) = 1.3905$$

$$\sum((\mathrm{d} - \mu(\mathrm{d}))\,(\mathrm{p} - \mu(\mathrm{p}))) = -252.05$$

$$\mathrm{r} = \frac{\sum_{i=1}^{n}(A_i - \mu(A))(B_i - \mu(B))}{\sqrt{\sum_{i=1}^{n}(A_i - \mu(A))^2}\sqrt{\sum_{i=0}^{n}(B_i - \mu(B))^2}} = \frac{-252.05}{\sqrt{232\,710}\sqrt{1.3905}} \approx \frac{-252.05}{569.232} \approx -0.44$$

So, we found out that there is a Pearson correlation of -0.44 between these two attributes, which could also be called a moderately strong anti-correlation. For more on the interpretation of correlation coefficient results, see the end of this handbook section.

Correlation matrix and correlation heatmap

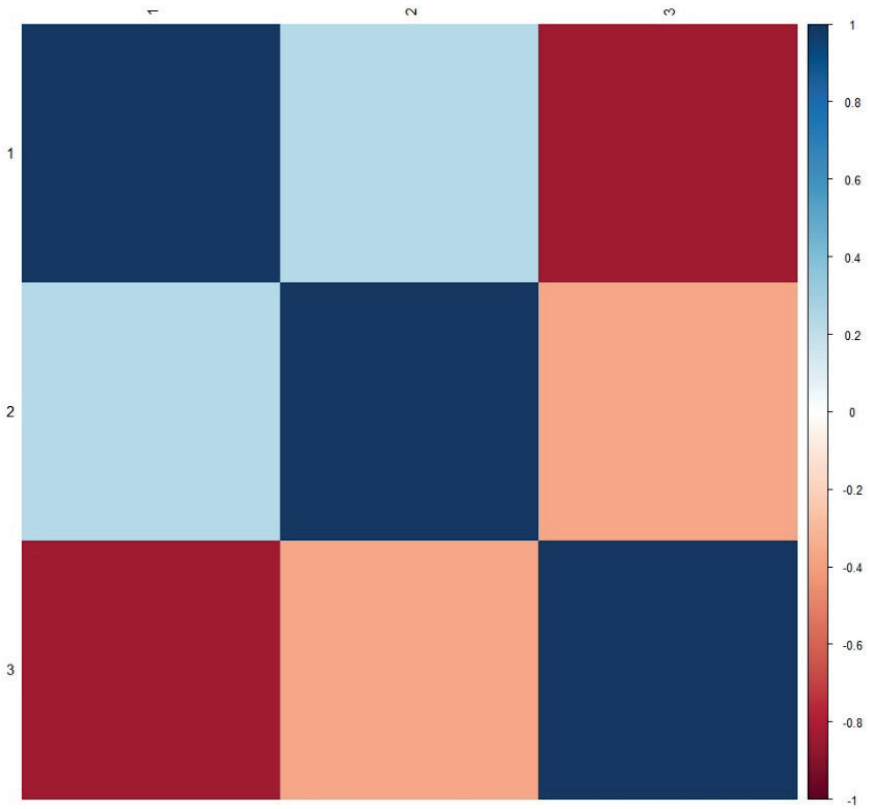
Datasets rarely contain only two attributes, which results in the need to measure correlation coefficients between all its attributes. For these purposes, we use a **correlation matrix** - a table containing a correlation coefficient measured between all possible pairs of attributes in the dataset. In the table below, we see the correlation coefficient measured between the attributes A_1, A_2, A_3 (correlation matrix), while we can see that $r(A_1, A_2) = 0.238$, $r(A_1, A_3) = -0.834$, and so on.

[1] For the lack of space in the presented table, we use d for distance and p for the price of the kebab.

	A_1	A_2	A_3
A1	1	0.238	-0.834
A2	0.238	1	-0.362
A3	-0.834	-0.362	1

This matrix has two **natural properties** - it is symmetric along the diagonal, and this diagonal always contains the values of the correlation coefficient equal to 1 – the correlation of the attribute A_1 with itself is always $r(A_1, A_1) = 1$ regardless of the method used, which is also natural since the value of attribute A1 is fully dependent on the value of attribute A_1.

Such a correlation analysis method is suitable only for a certain number of attributes in the studied dataset. Obviously, for a dataset containing dozens of attributes, such a matrix would be confusing and hard-to-read. Therefore, a so-called correlation heatmap or correlation plot often replaces i**t**. For the correlation matrix present above, the heatmap can be constructed as follows:

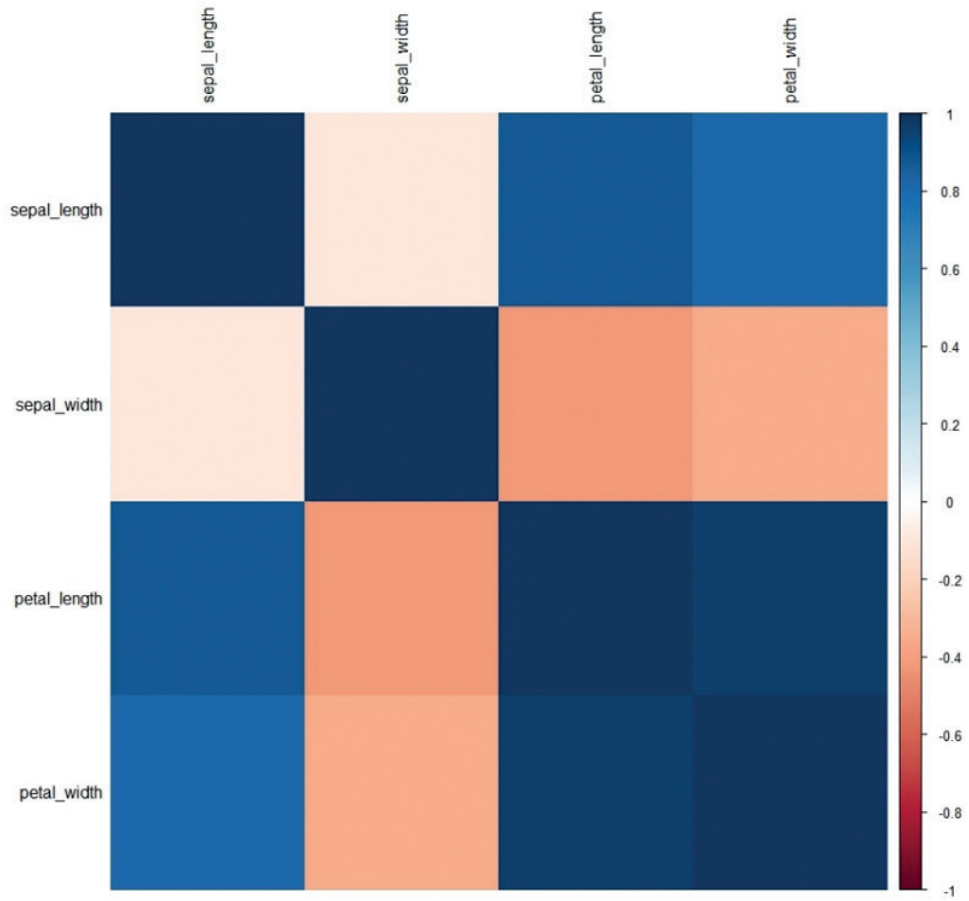


Such a correlation heatmap is just a simple projection of the correlation matrix into a color grid, in which the color of the field is defined by the value of the correlation coefficient for the given pair of attributes. For better readability, the scale (right) containing the interval of possible values for the correlation coefficient is indicated in the correlation heatmap. Instead of looking for numbers close to the extremes of the interval [1,–1] in the correlation matrix, in the correlation heatmap, we look for dark red or dark blue grid fields that indicate the same property and are easier to identify for most people.

Example: Let us have the Iris dataset described in Appendix A of this handbook. This dataset contains five attributes measured on 150 entities, four of which are numerical attributes, and the fifth one contains a linguistic value indicating the class for the given entity. As part of the correlation analysis, it is impossible to work with linguistic attributes, so we will consider only the dataset of size 150×4. The Pearson correlation matrix of the (somewhat cropped) Iris dataset contains values as follows:

	sepal length	sepal width	petal length	petal width
sepal length	1	-0.1093692	0.8717542	0.8179536
sepal width	-0.1093692	1	-0.4205161	-0.3565441
petal length	0.8717542	-0.4205161	1	0.9627571
petal width	0.8179536	-0.3565441	0.9627571	1

Of course, this dataset does not contain a large number of attributes, so creating a correlation heatmap is not necessary for the act of correlation analysis. However, despite this, we present the heatmap as a demonstration of the projection of correlation coefficient values into the color scale presented in the correlation heatmap.



Interpreting results of correlation coefficients

The correlation coefficient shows how much it is possible to predict the values of attribute A_2 in the selected data sample based on attribute A_1. The closer the value of this coefficient is to the extremes of the considered interval (that is, to the value 1 or -1), the more suitable the given attribute A_1 is for predicting the values of the optional attribute A_2.

Obviously, a value of **0 is the worst case** for the correlation between two attributes. In such a case, there is no relationship between the values of these attributes that could be used in the case of building mathematical models on the dataset we are working with.

The literature differs slightly in its view of the level of acceptability of correlation values - except that the higher the correlation, the better. In general, we speak of two attributes as strongly correlated when the value of the correlation coefficient measured between them reaches values **higher than 0.8**. A strong anticorrelation exists between the two attributes if the correlation coefficient reaches a value **lower than -0.8**. This limit of acceptability of the prediction potential can be **relaxed closer to the values of 0.7 or -0.7**, but more is not recommended.

4.3 Exploratory data analysis and data visualization

Exploratory Data Analysis (commonly referred to as EDA) is a data analysis method that explores data to find patterns and trends in a given population or sample. In its most basic form, this type of analysis is performed by visual exploration of data. Before the visualization itself, it is necessary to take several steps that will be beneficial in the further search for knowledge hidden in data:

- **Familiarization with the dataset** - it is necessary to be able to answer several questions related to the given set of data before we analyze it:
  - **Who compiled the dataset, when, and why?** This point is essential from the point of view of relevance, topicality, and usefulness of the dataset. If the dataset was compiled by experts in the field, it would be more relevant than if it was compiled by a beginner who measured data on a cheap consumer-grade sensor. If the dataset was compiled 93 years ago, it is possible that the data would not be up to date, the measurements would be less accurate than we would be able to measure today, etc. The dataset is also built with a specific purpose and is, therefore not universal (it may not be suitable for all tasks).
  - **How big is this dataset?** By the size of the dataset, we mean the number of entities and attributes measured in the dataset. In case we have a dataset too big to work with comfortably (see section 1), we need to select a sample from it based on the principles mentioned in the previous section of this handbook. The opposite problem - a too-small dataset - is much bigger. However, some algorithms can work with small datasets and approaches that generate new entities based on existing ones, called oversampling.
  - **What is the composition of the datasets?** This point is closely related to the reason for compiling the dataset. It is important to go through all the dataset attributes and understand their purpose. It is also necessary to focus on whether the data recorded in the given attribute is numerical or categorical and in what range the values of the individual attributes move.
- **Computation of summary statistics** - for each attribute, it is advisable to compile basic summary statistics. Recommended values are extreme (min, max), median or average, standard deviation, and others. This is a very important and informative step in which we obtain the mean values of the gi-

ven attributes, their smallest and largest values, and we can further describe the individual attributes by aggregation.

- **Performing correlation analysis** - for any dataset, it is advisable to compile the matrix or heatmap of correlation coefficients. This matrix measures the correlations between the values of all the attributes of the dataset and thus shows us how difficult it will be for us to build mathematical models on the given set of data. Correlation analysis also helps in the identification of attributes and subsets of dataset fit for visualization.

The next step of exploratory data analysis is the actual visualization of the dataset. However, we are talking about effective data visualization based on several principles presented in the next part of this handbook.

Effective data visualization

We call data visualization effective in the case that:

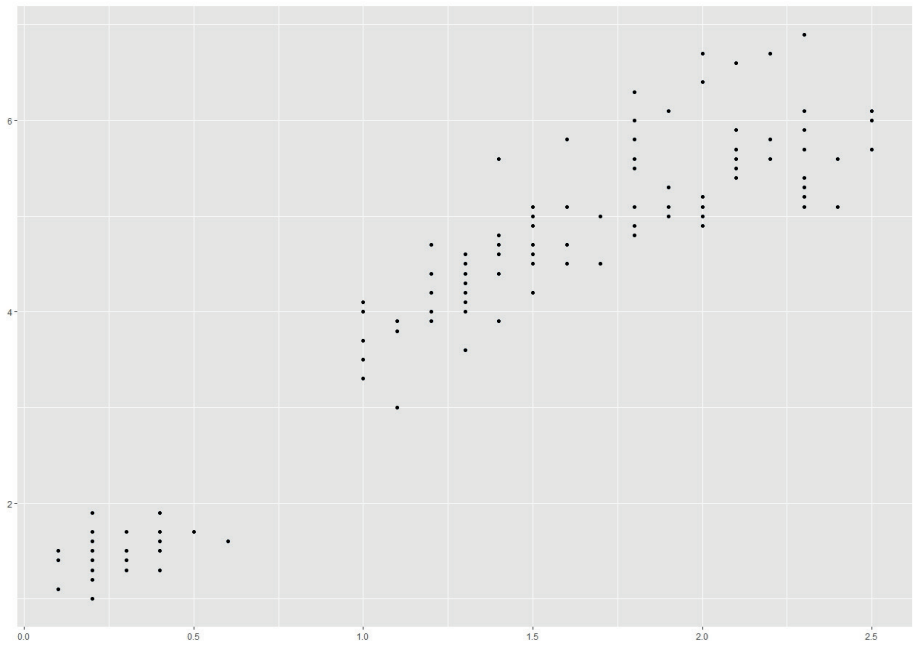
- we visualize the correct data,
- the right way,
- using the correct graph type.

These three points are quite natural. Data that could be called suitable for visualization from the point of view of data analysis are those that carry some knowledge - most often predictive potential. As we already know from the previous section of the handbook, we can easily search for predictive potential in datasets using correlation analysis methods. Therefore the parts of the dataset suitable for visualization will be those where we have identified strong correlations or anticorrelations between attribute values (see section Interpreting results of correlation coefficients).

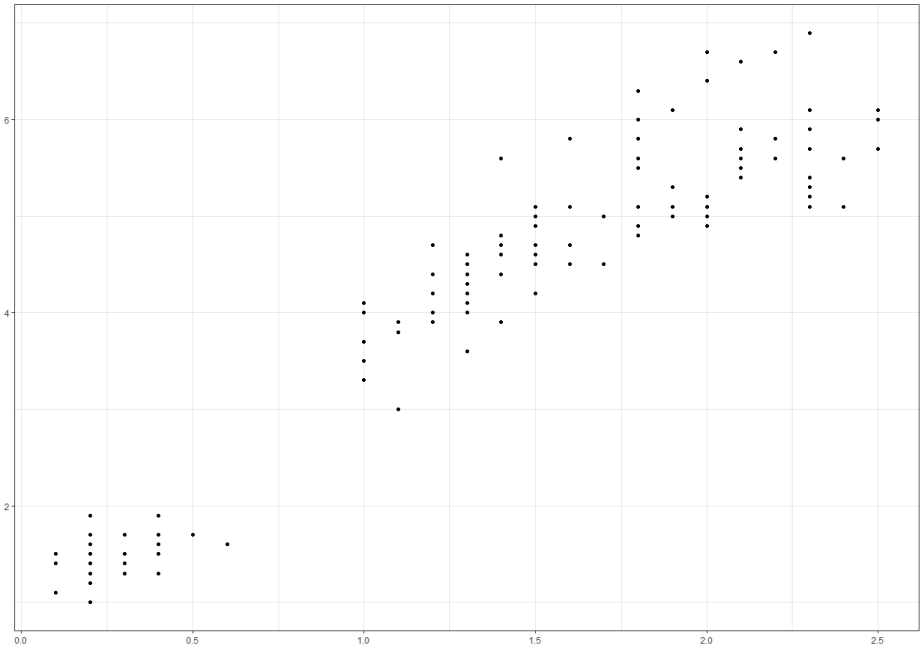
In our case, by the **correct way of data visualization**, we mean the elimination of two relatively common problems present in the visualization of datasets:

- **Maximization of the ratio between used color and data** - since we want to visualize the data, ideally, the graph will contain a minimum of other graphic elements (for example, a background color, a distinctive grid, and so on). Maximizing the ratio of color and data is essential, especially when visualizing large sets of points that can potentially merge or be rendered by very small points (or another type of objects). The figure below contains a standard point graph drawn in the R language (subfigure a) and a modification of this graph to make the data more visible.

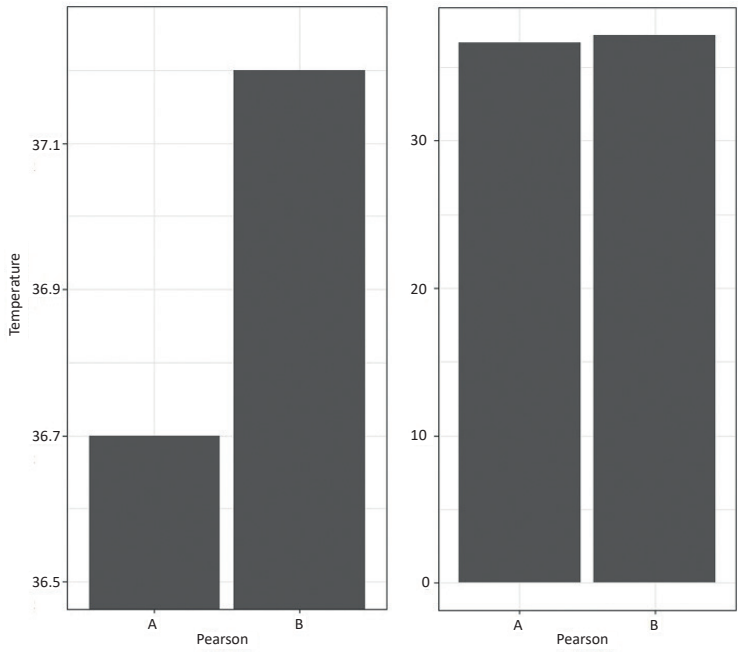
a)



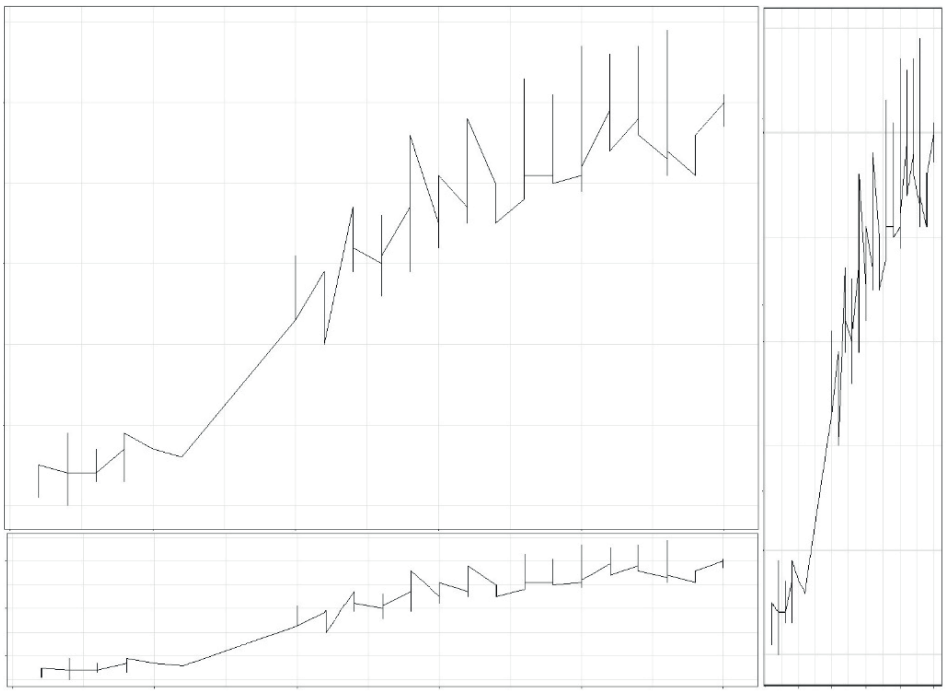
b)



- **Elimination of distortions** - in the visual presentation of data and their subsequent interpretation, distortions often occur due to the incorrect setting of the axes or due to the distortion of the image file itself. The figures below present both of these problems. The first bar graph compares the temperature of two people (A and B). Note that the values on the left graph appear to be significantly different, while the values on the right are very similar, even though these are two identical measurements presented with different axis settings. In the left figure, we present the y-axis only from the value of 36.5 to 37.3 degrees. On the right graph, we present the values of y from 0 to 40 degrees. This is a typical factor of confusion in data visualization.



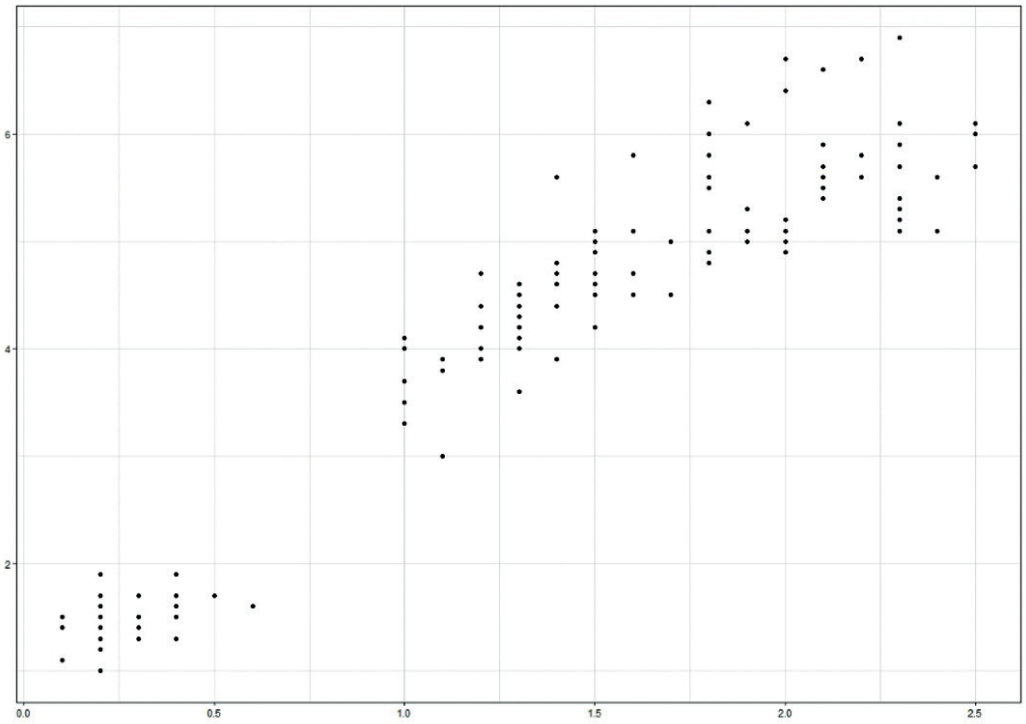
The second distortion problem is the distortion of the image itself. In the image below, we can see one graph presented in three aspect ratios of the graph. It is clear that the image on the bottom and right are inappropriate due to the distortion of the actual shape of the trend presented by the given graph, and therefore the ideal ratio for this data will be the upper, left graph - close to the standard image ratio of most modern projectors - 16:9.



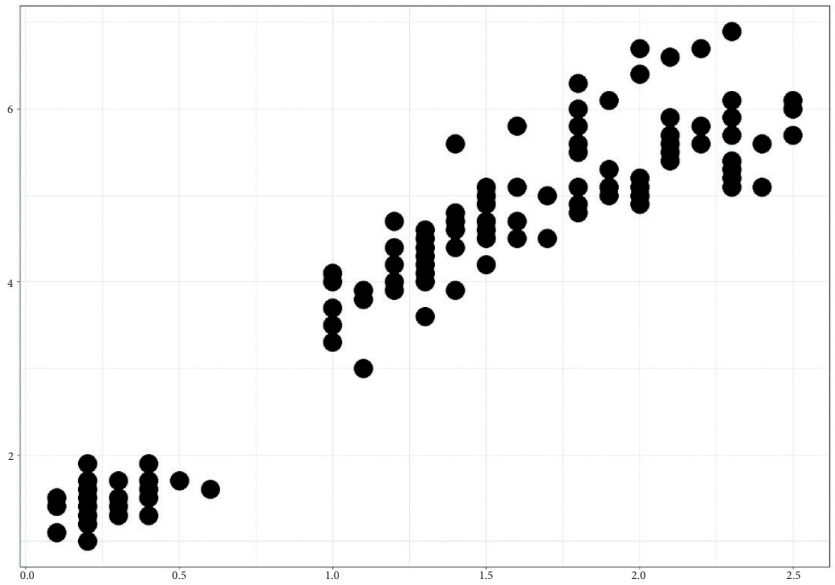
The last very important element of the visualization's effectiveness is the correct type of graph. Generally, four types of graphs are popular - point, line, pie, and bar graphs. Each graph type is suitable for different purposes and has different advantages and disadvantages. This handbook section will focus only on the two most common data visualization methods - point graphs and line graphs.

Point graphs

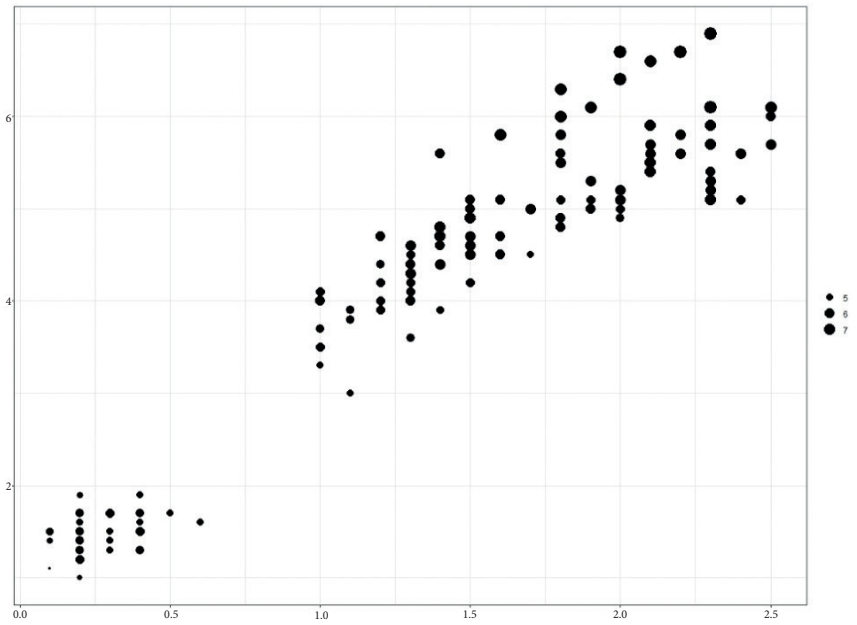
Point graphs are used to visualize the relationship between two (or more) attributes of a dataset using points. The standard way of visualization is to compare the values of two attributes in a plane:



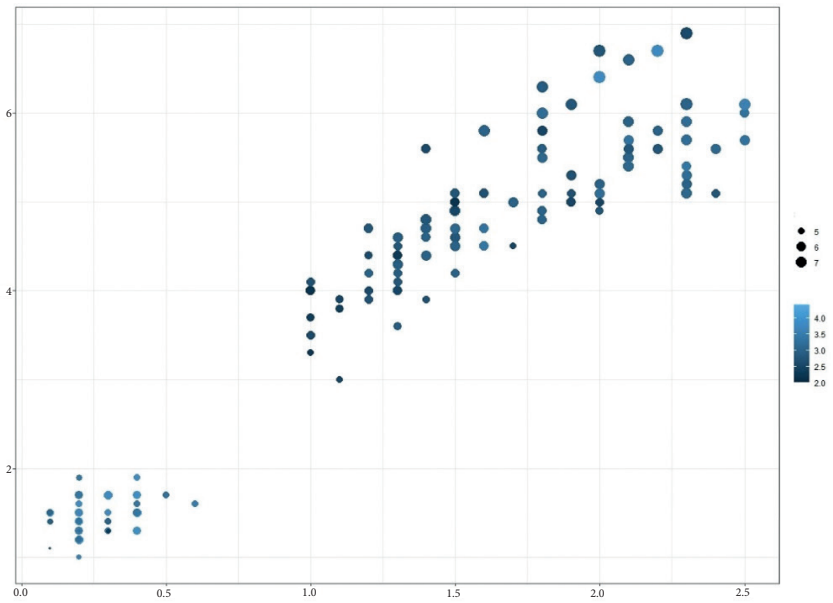
With such an approach to data visualization, we must pay attention to the point size in the graph. The image below shows the oversaturation of the graph caused by large point size; a similar problem occurs with a large number of points that are located close to each other.



However, point size can be used in the context of data visualization to convey additional information. If we set the size of the point directly proportional to the size of the third attribute in the dataset, we can visualize the relationship between the three attributes of the dataset.

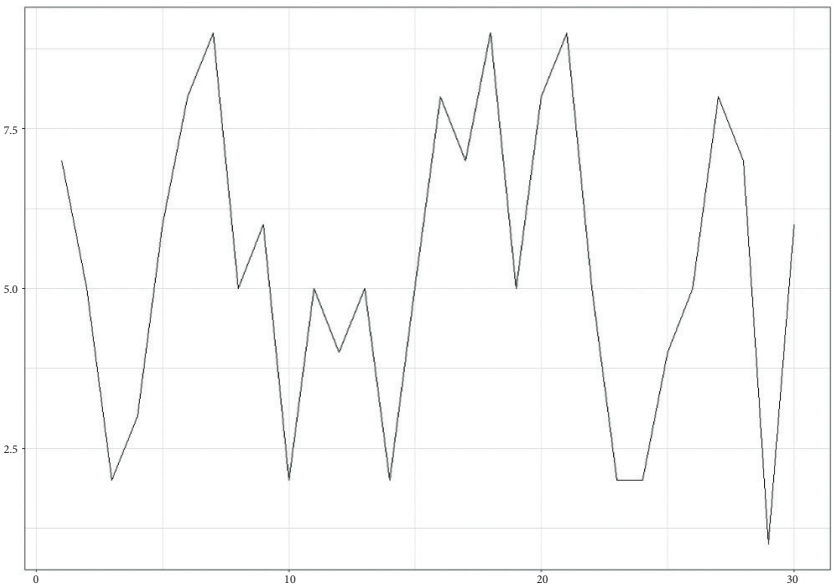


We can extend this concept with other properties of the points, for example, their color (listed below). Thus we can achieve the visualization of the relationship between four attributes of the dataset. However, this method of visualization is difficult to use with a large number of points or attributes. In general, it is not recommended to visualize more than three or four attributes in a plane.

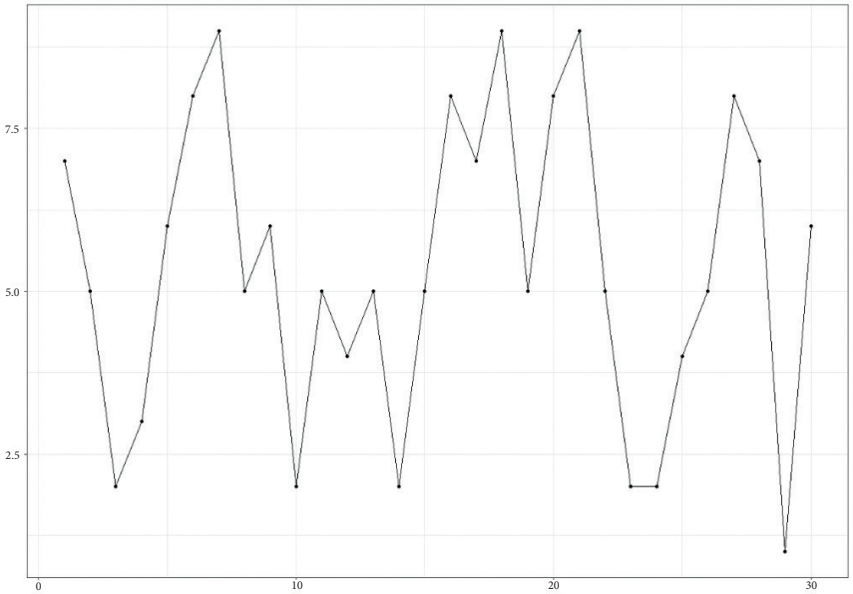


Line graphs

Line graphs are used to visualize the course of the value of one attribute over time or to visualize the fluctuation of the value of an attribute depending on another attribute. It is important to note that the lines present in line graphs are an approximation of points - the transformation of discrete data points into continuous lines and, therefore might be inaccurate in some places. Compared with point graphs, line graphs are hard-to-use for categorical (linguistic) data.



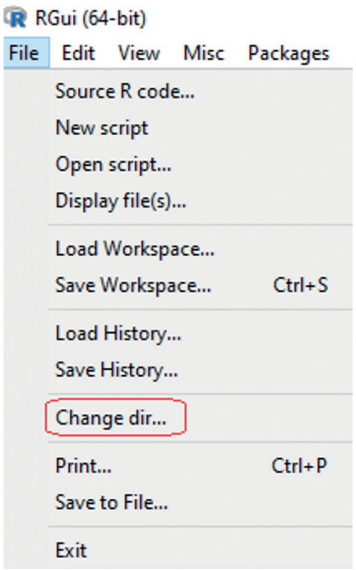
Since lines are approximations of data points, it is recommended to visualize the line and the points the line is based on. This lowers the ambiguity of the correctness of attribute values.



4.4 Exploratory data analysis in practice

This part of the handbook is focused on the practical application of exploratory data analysis methods in the R language. The examples given below were applied in version *R 4.3.0,* but all of these concepts and commands are present in all versions of all programming tools suitable for data analysis.

First, we need to change the working directory for our session to the directory where our structured data is stored - this can be done with the use of the top bar of the program, where: *File → Change dir → set the working directory*. For the purposes of exploratory data analysis method presentation, we will use the Iris dataset described in *Appendix A*.



After changing the working directory, we can start loading the dataset to the R. This operation can be performed in several ways. We present (from our point of view) the simplest one - *read.table* and *read.csv* commands, which work similarly for different types of input files. The command in the form *read.csv* is used for *.csv* files, the most common, structured input for data analysis tools. The form *read.table* is usable for input in the format *.txt* or *.data*.

```
read.table("title", header=T/F, sep="symbol")
read.csv("title", header=T/F, sep="symbol")
```

where *title* is the name of the file where our data is stored along with the file extension, the *header* part of the command indicates whether the input data file has a header (*T* for true) or not (*F* for false), and the *sep* part of the command expects the character by which attribute values in the file are separated.

To be able to use the loaded dataset further in the program, we need to save it under a selected title, for example, *title_of _data*:

```
title_of_data <- read.table("title", header=T/F, sep="symbol")
```

An example of such loading of a data file stored under the title *our_data.data* can look as follows. In the second given command, we save our dataset under the title *data*:

```
read.table("our_data.data", header=T, sep=",")
data <- read.table("our_data.data", header=T, sep=",")
```

Exploratory Data Analysis - Step 1 - Familiarization with a given dataset

As part of getting to know the dataset, we can perform several very simple operations. The first of them is to list the entire dataset in the console of the R tool using *title_of _data*. However, this is impractical for large data sets containing thousands of entities. Therefore we present the second version of the command to list the entities of the dataset - head, which lists the defined number of entities from the beginning of the dataset to the console.

```
title_of_data
head(title_of_data, number_of_entities)
```

An example of this concept could be a listing of the entire dataset stored under the name data or a listing of the first five entities of this file.

```
data
head(data, 5)
```

The output of this command in the R language will be a pseudo-table in the following format:

```
> data <- read.table("iris.data", header = T, sep = ",")
> head(data, 5)
  sepal_length sepal_width petal_length petal_width       class
1          5.1         3.5          1.4         0.2 Iris-setosa
2          4.9         3.0          1.4         0.2 Iris-setosa
3          4.7         3.2          1.3         0.2 Iris-setosa
4          4.6         3.1          1.5         0.2 Iris-setosa
5          5.0         3.6          1.4         0.2 Iris-setosa
```

After a basic familiarization with the dataset, its attributes, and values in them, we can proceed to the computation of measures of centrality and variability. All these functions are derived from the English version of the name of the individual functions (e.g., *sd* for standard deviation), and their input is only one of the attributes of the dataset written in the form

title_of _data$title_of _attribute.

The most versatile of these commands is the summary function, which measures the *minimum, 1st quartile, median, mean, 3rd quartile,* and *maximum* for all attributes from the dataset.

```
mean(title_of_data$attribute_title)

median(title_of_data$attribute_title)

min/max/sum(title_of_data$attribute_title)

sd(title_of_data$attribute_title)

summary(title_of_data)
```

An example of such an analysis of statistical properties present in the data is the use of the following commands:

```
summary(data)

sd(data$attribute_title)
```

The output of these functions executed on the Iris dataset consists of the following set of values:

```
> summary(data)
  sepal_length    sepal_width     petal_length    petal_width          class
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Length:150
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Class :character
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Mode  :character
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> sd(data$sepal_length)
[1] 0.8280661
```

Exploratory Data Analysis - Step 2 - Correlation analysis

As mentioned in previous sections of this handbook, correlation analysis is one of the most essential parts of exploratory data analysis. The basic form of the command for correlation analysis is the computation of the correlation coefficient between two dataset attributes using the *cor* function. In the command syntax presented below, we can see that the type of correlation coefficient that we want to compute for the data can be defined using the parameter *method = type_of _correlation*. The Pearson correlation coefficient is the default used in this command.

```
cor(title_of_data$attribute_title_1, title_of_data$attribute_title_2)

cor(title_of_data$attribute_title_1, title_of_data$attribute_title_2,
method = “pearson”)

cor(title_of_data$attribute_title_1, title_of_data$attribute_title_2,
method = “spearman”)
```

However, as part of the dataset analysis, we want to examine all the relationships between all the attributes of the dataset, and therefore we can create a correlation matrix:

```
cor(title_of_data)

cor(title_of_data, method = “spearman”)
```

Correlation analysis of the Iris dataset can be done with the use of the following simple commands:

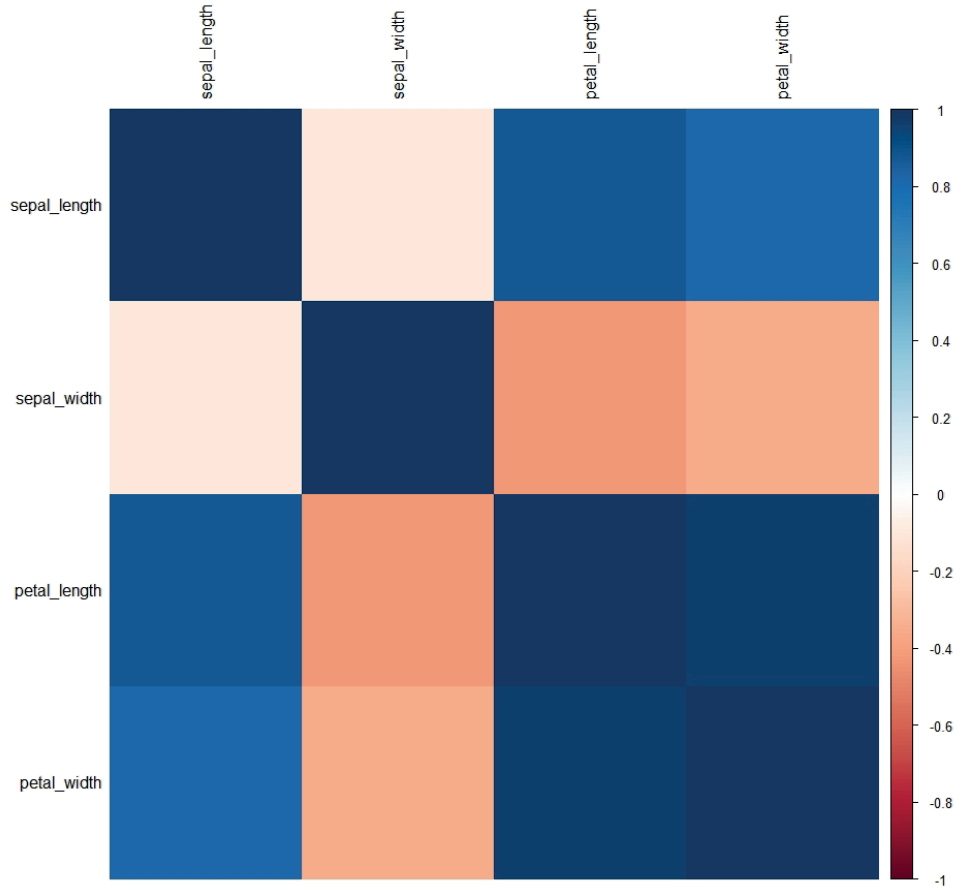
```
cor(data[, 1:4])

cor(data[, 1:4], method = “spearman”)
```

Note: Since the Iris dataset contains one attribute whose values are linguistic (attribute class) and the correlation matrix comprises only numerical values, the function cor must have only the first four (numeric) attributes as input. We achieve this by selecting columns 1:4 from a dataset called data: data[, 1:4].

```
> cor(data[,1:4])
             sepal_length sepal_width petal_length petal_width
sepal_length    1.0000000  -0.1093692    0.8717542   0.8179536
sepal_width    -0.1093692   1.0000000   -0.4205161  -0.3565441
petal_length    0.8717542  -0.4205161    1.0000000   0.9627571
petal_width     0.8179536  -0.3565441    0.9627571   1.0000000
> cor(data[,1:4], method = "spearman")
             sepal_length sepal_width petal_length petal_width
sepal_length    1.0000000  -0.1594565    0.8813864   0.8344207
sepal_width    -0.1594565   1.0000000   -0.3034206  -0.2775111
petal_length    0.8813864  -0.3034206    1.0000000   0.9360034
petal_width     0.8344207  -0.2775111    0.9360034   1.0000000
```

As mentioned in section 4.2, using a correlation heatmap for large datasets is advisable. We need to install a package of functions in the R language to use this visualization method, which contains a correlation heatmap visualization function called *corrplot*. After installing this package, we load it using the *require* function and then create a correlation heatmap:

```
install.packages("corrplot")
require(corrplot)
corrplot(cor(data), method = "color")
```



The correlation matrix and heatmap for the Iris dataset tell us about the relationships that can be used in further data analysis. Specifically, we are interested in all relationships between attributes where any type of correlation coefficient was higher than 0.8 or lower than -0.8:

- ρ(sepal_length, petal_length) ≈ 0.87
- ρ(sepal_length, petal_width) ≈ 0.82
- ρ(petal_length, petal_length) ≈ 0.94

These relationships between attributes are worthy of visualization.

Exploratory Data Analysis - Step 3 - Data visualization

After analyzing the correlations, we are ready to visualize pairs of attributes where we have noticed a strong correlation or anticorrelation. However, before the visualization itself, we need to install a package used for visualization purposes:

```
install.packages("ggplot2")
require(ggplot2)
```

Package *ggplot2* is one of the most popular packages used in data visualization, which contains functions for drawing point graphs, line graphs, bar graphs, and many more. In this section of the handbook, we select some simple examples of these functions.

Point graphs

The most basic and, at the same time, the most powerful visualization of data is the point graph. Within the R language and the *ggplot2* package, we use the *ggplot* function to construct any type of graph while the function expects several values as input. For the simplest of graphs, these inputs are:

Title of the dataset we are working with (in our case, this title is *data*).

- Section *aes* derived from the English word aesthetics and expects information about at least one axis. This information is presented in the form axis_title *(x or y) = title_of _attribute* represented by the axis.
- Type of graph.

In general, the syntax of commands for point graphs contains the assignment of two attributes to the graph's axes and section of the command + *geom_point()*. The generalized syntax for this type of command is presented below:

```
ggplot(title_of_data, aes(x = title_of_attribute_1, y = title_of_
attribute_2)) + geom_point()
```

We can change the color of the points using the extension of the + *geom_point()* command section by adding a second *aes* section valid only for the points themselves - namely + *geom_point(aes(color = "color name"))*. As an alternative to specifying a color, we can modify the color of the points within one graph by specifying the attribute's name from the dataset we are working on within the section of the + *geom_point()* command instead of the color name. In this way, we achieve visualization in two dimensions (attributes) with an additional dimension marked by the color of the points, which is changed based on the values of the chosen attribute. To adhere to the principles of effective data visualization presented in the previous subsection, we add the + *theme_bw()* option to the end of the command, which ensures a white background under the graph itself, thus maximizing the ratio between data and color used in the graph.

```
ggplot(title_of_data, aes(x = title_of_attribute_1,
y = title_of_attribute_2))) + geom_point(aes(color = “color”))
+ theme_bw()
ggplot(title_of_data, aes(x = title_of_attribute_1, y = title_of_
attribute_2))) + geom_point(aes(color = title_of_attribute_3)) + theme_bw()
```

A simple example of this approach can be done as presented below. We also introduce two additional concepts:

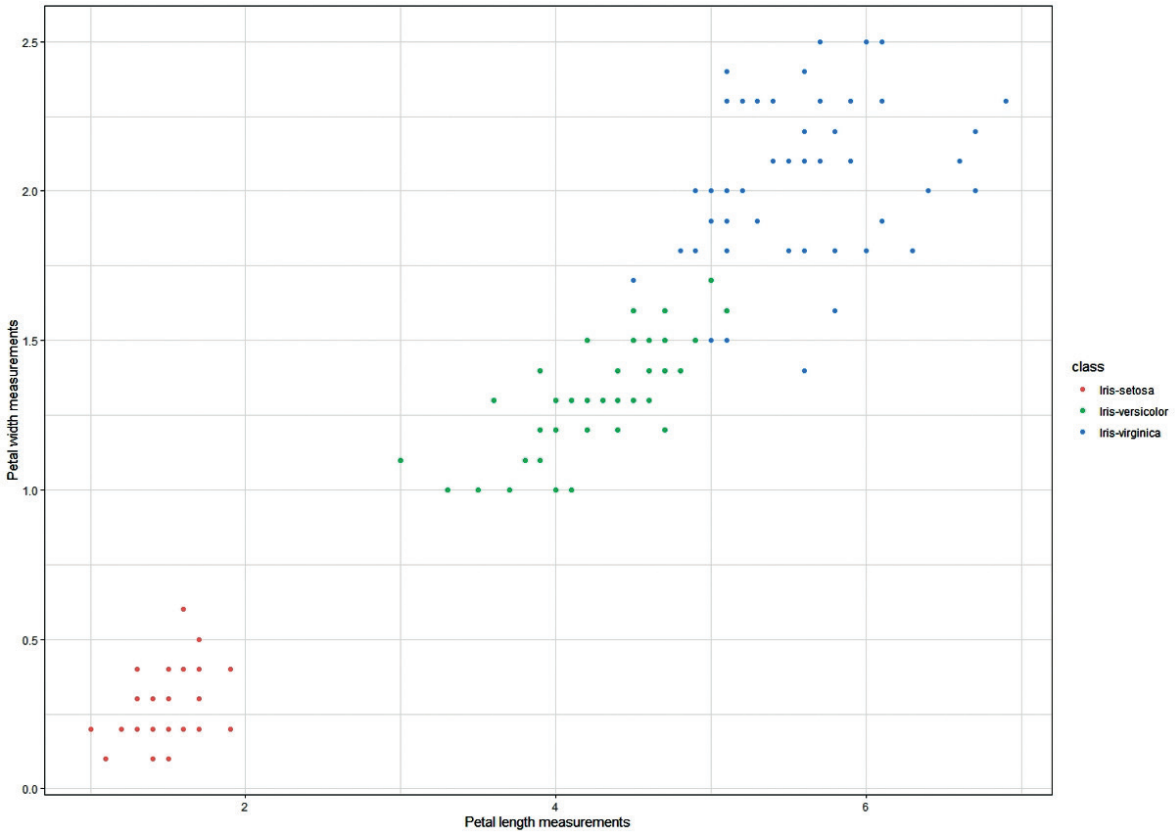
- The created graph can be saved under a selected name, for example, graph1, as shown below.
- We can add other parts of the command to the graph saved in this way. In the example below, we use + *xlab()* and + *ylab()* to add x and y axes labels. The graph is then visualized by calling its name in the console.

```
graph1 <- ggplot(data, aes(x= atr_1, y = atr_2))
+ geom_point(aes(color = class) + theme_bw()
graph1 <- graph1 + xlab(“X parameter”) + ylab(“Y parameter”)
graph1
```

So, the following code for the Iris dataset

```
> require(ggplot2)
> graph1 <- ggplot(data, aes(x = petal_length, y = petal_width)) + geom_point(aes(color = class)) + theme_bw()
> graph1 <- graph1 + xlab("Petal length measurements") + ylab("Petal width measurements")
> graph1
```

produces the following figure:



Line graphs

Another one of the frequent type of graphs is the line graph, used to visualize the course of the value of one attribute over time or to visualize the fluctuation of the value of an attribute depending on another attribute. The syntax for the line graph command in the ggplot2 package is not significantly different from the previous examples presented in this section of the handbook. The only difference is the type of geometry used when drawing the graph - in the case of line graphs, it is + *geom_line()*. Using the *linetype* option in the section of the *geom_line()* command, we can also change the type of line that is used when plotting data.

```
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) +
geom_line()
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line
(linetype = “dashed”)
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line
(linetype = “twodashed”)
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line
(linetype = “dotted”)
```

Similarly to point graphs, we can easily change the color of geometry - in this case, line - using option color, which can be combined with all of the linetypes in the graph.

```
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line(color = “color”)

ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line(linetype = “type”, color = “color”)
```

Since a line graph is always an approximation of data points, it is appropriate to visualize the points themselves alongside the line graph. This can be done very simply by a combination of both line and point geometries as follows:

```
ggplot(title_of_data, aes(x=attribute_title_1, y=attribute_title_2)) + geom_
line() + geom_point()
```

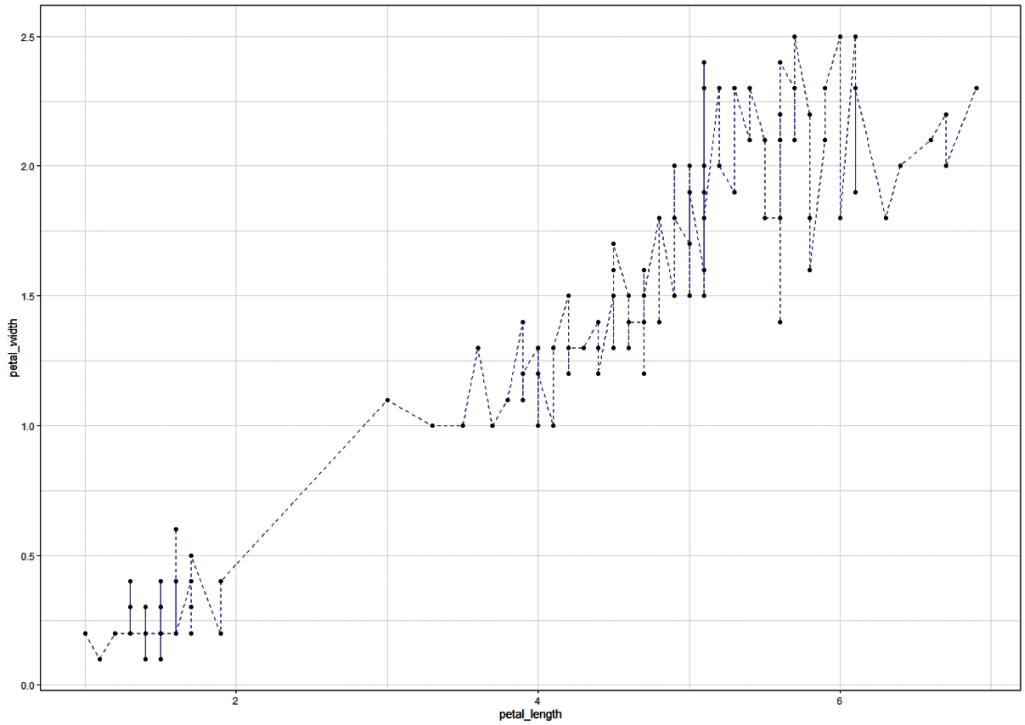
Understandably, a combination of all of these options is possible:

```
ggplot(data,   aes(x=attribute_title_1,   y=attribute_title_2))   +   geom_
line(linetype = “dashed”, color = “blue”) + geom_point()
```

So, the following code for the Iris dataset:

```
> ggplot(data, aes(x = petal_length, y = petal_width)) + geom_line(linetype = "dashed", color = "blue") +
+ theme_bw() + geom_point()
```

generates the following figure:



The true power of line graphs is in their ability to visualize the comparison between the values of an attribute and a set of attributes - in other words, the possibility of drawing more than one line. In the example of syntax below, we can see that the core *ggplot* function contains only one attribute (the one placed on the x-axis), and we use the y-axis to visualize values of attribute2 and attribute3. This visualization is done by separate + *geom_line()* sections of code. We also present several combinations of the options above with this approach.

```
ggplot(title_of_data, aes(x=attribute_title_1))
+ geom_line(aes(y= attribute_title_2))
+ geom_line(aes(y= attribute_title_3))

ggplot(title_of_data, aes(x=attribute_title_1))
+ geom_line(aes(y= attribute_title_2), color = “color”)
+ geom_line(aes(y= attribute_title_3), color = “color”)

ggplot(title_of_data, aes(x=attribute_title_1))
+ geom_line(aes(y= attribute_title_2), linetype = “type”, color = “color”)
+ geom_line(aes(y= attribute_title_3), linetype = “type”, color = “color”)
```

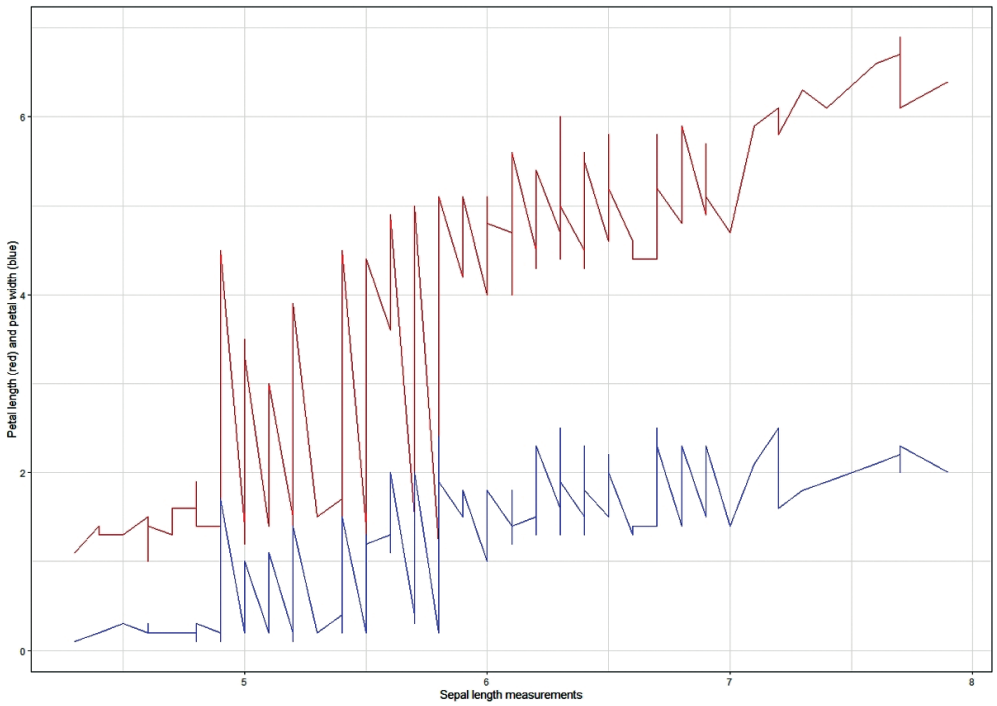
Let us have the Iris dataset in which we measured strong correlations between three attributes - sepal length, petal length and petal width. Fluctuation of petal length and petal width depending on sepal length can be visualized using two lines, which will be separated by their type or color. In our case:

- fluctuation of values of petal length based on sepal length is visualized with the use of red line,
- fluctuation of values of petal width based on sepal length is visualized with the use of blue line.

```
ggplot(data, aes(x=attribute_title_1)) + geom_line(aes

(y= attribute_title_2), linetype = “dotted”, color = “red”)

+ geom_line(aes(y= attribute_title_3), color = “blue”)

So, the following code for the Iris dataset
```

```
> ggplot(data, aes(x = sepal_length)) + geom_line(aes(y = petal_length), color = "red") +
+ geom_line(aes(y = petal_width), color = "blue") + theme_bw() + xlab( "Sepal length measurements") +
+ ylab("Petal length (red) and petal width (blue)")
```

produces the following figure:



Such data visualization can be used to look for trends and patterns in the data, which can then be used in more complex approaches to data analysis presented in other sections of this handbook. Exploratory data analysis has one shortcoming - it is hard to use on really Big Datasets which require dimensionality reduction and some other techniques to analyze properly.

CHAPTER 5

FUZZY SETS

This part of the handbook was written by Alžbeta Michalíková from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

In everyday life, we often use vague expressions such as a young person, a bit of salt, slightly to the right, strong wind, high temperature, low price… These expressions do not have precise boundaries. They are not clearly defined. We can call them **vague** or **fuzzy**. The first idea of mathematical modelling by fuzzy concepts could be found in the paper of Lotfi A. Zadeh [1]:

ZADEH, L. A.: Fuzzy sets. Information and control. Volume 8, pp. 338-353, 1965.

Using fuzzy sets is referred to as **operating with the natural language** (operate not only with numbers but also with other terms of human speech). At the same time, it is understandable that different people perceive different concepts differently. The main part of the following text is based on the university textbook [2] **Fuzzy sets in Informatics** (in Slovak), which is intended for students of applied informatics at the Department of Computer Sciences, Faculty of Applied Sciences, Matej Bel University in Banská Bystrica, Slovakia.

Why use fuzzy logic?

- The idea of fuzzy logic is easy to understand,
- it is a flexible system that is tolerant of inaccurate data,
- it can work with the experiences of the experts,
- it can model a non-linear system of any complexity,
- it can be used in standard technical equipment.

Uses of fuzzy sets

- Expert systems,
- recognition and classification of objects,
- theory of control and regulation,
- database systems,
- mathematical modelling,
- recently – explainable neural networks.

Areas of application of fuzzy sets

Wherever the uncertainty is included in the calculation. They are often used in devices that do not represent expensive appliances from an economic point of view, e.g., **electronic household appliances** (washing machines, microwave ovens, vacuum cleaners, shavers, pressure gauges, ...). We can also find them in **complicated and economically** as well as **computationally intensive devices**, for example

- driving of the metro in Japan - (city Sendai – from 1988) [3],
- blast furnace control (temperature control which can be controlled more efficiently than with conventional regulators),
- managing nuclear power plants [4], ...

Example: Let us have an advertisement for a work position that requires that the age of the candidates is in the interval of 20-30 years of age. Let's describe this set!

What is the universe?

Can you describe this set by its characteristic function?

Can a person who will have their 31st birthday tomorrow answer this advertisement?

What is the universe?

This set is usually denoted by the letter X. The universe should be any interval with achievable values. For example, in our task, it could be X=(0,∞).

Can you describe this set by its characteristic function?

A characteristic function is a function that assigns the number 1 to those elements, which belong to the considered set, and, on the other side, it assigns the number 0 to those elements which do not belong to the considered set. This function is usually denoted by the letter χ. For our example, the function has the following notation:

$$\chi_A: \mathbb{X} \to \{0, 1\} \qquad \chi_A(x) = \begin{cases} 1, & \text{if } 20 \le x \le 30, \\ 0, & \text{if } 0 \le x < 20 \text{ or } x > 30. \end{cases}$$

Can a person who will have their 31st birthday tomorrow answer this advertisement?

NO! - because when someone reads their answer, they will no longer meet the required condition.

Example: Let's have a similar situation: In an advertisement, there is a requirement that the company is looking for young people.

Has the situation changed from the previous example?

What is the universe?

How can we describe this set?

Can a person who will have their 31st birthday tomorrow answer to this advertisement?

Has the situation changed from the previous example?

YES! – the set of young people represents the **fuzzy set**. There is no sharp boundary for the age of the people who belong to this set!

What is the universe?

The universe of fuzzy set should be any interval with achievable values. It could be the same set as it was at the classical set, e. g. X=(0, ∞).

How can we describe this set?

For example, a 20-year-old person is young for sure. Therefore, it assigns a degree of youth equal to 1. Similarly, a 30-year-old person can be considered young for sure. Therefore, it assigns a degree of youth equal to 1, but to a 35-year-old, we could assign a degree of youth 0.8, ... To describe fuzzy sets, we use so-called **membership functions**. They are denoted by the letter μ. By using this function, we need to assign some value from the unit interval (e. g. from interval (0,1) to each element of the universe). Firstly, let´s look at the values of ages, which we considered as exactly young person age. As an example, these values could be from the interval (0,30). The membership function assigns a value equal 1 to these values. Now, let´s look at the values of ages, which we considered as exactly not young person age. An example of such values could be values greater than 55. To these values, the membership function assigns a value equal 0. For the values from the interval interval (30,55) we expect that the values of membership to the set of young persons will sequentially descend from 1 to 0 (see Figure 1).

Denote the fuzzy set of young people by B. Prescription of a described function is

$$\mu_B : \mathbb{X} \to \langle 0, 1 \rangle \qquad \mu_B(x) = \begin{cases} 1, & \text{if } x \in \langle 0, 30 \rangle, \\ \frac{1}{25}(55 - x), & \text{if } x \in (30, 55\rangle, \\ 0, & \text{if } x > 55. \end{cases}$$

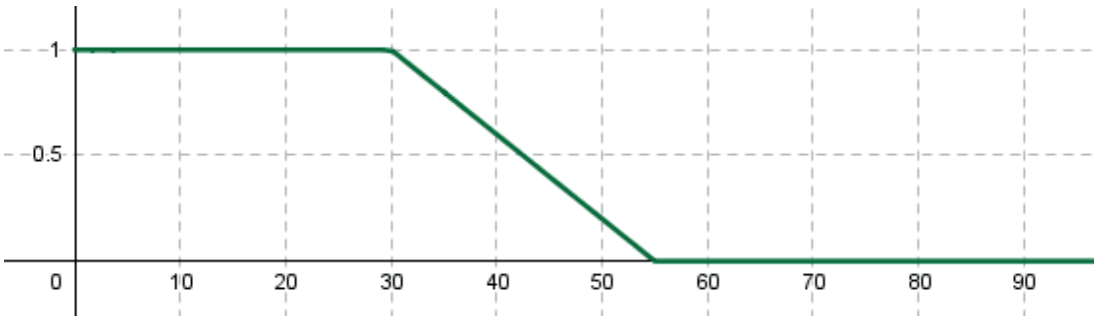


Figure 1: Membership function of the fuzzy set of young people

Remarks:

The terms fuzzy set and membership function are often considered equivalent.

The value, which is assigned to a specific input value, is called **the degree of membership** or **membership degree**.

Let the fuzzy set determined by the formula (1) describe the set of young people. Can we assign the membership degree to people who are 20, 35, and 45 years old? By using formula (1) we get

μ_B (20) = 1, i.e., 20 years old person is young for sure,

μ_B (35) = 0.8, i.e., 35 years old person is young whit a degree 0.8,

μ_B (40) = 0.6, i.e., 40 years old person is young whit a degree 0.6.

Can a person who will have their 31st birthday tomorrow answer this advertisement?

YES! – because his degree of membership function to the fuzzy set μ_B equals 0,96 (since $\mu_B(31)$ = 0,96). This value represents a high degree of membership in the fuzzy set of young people.

Remark:

There is a number of types of membership function of fuzzy sets. We will show some of them in the next example.

Example: Let's model the fuzzy set C of real numbers, representing "approximately 7".
What is the universe?
How can we describe the properties of this set?
With the use of this term "approximately 7" we could imagine sentences such as

It is approximately 7 degrees of Celsius outside.
or
I spent approximately 7€ in the store.

What is the universe?

As a universe, we usually consider the greatest possible set. In this example, it could be the whole set of real numbers, e. g. .

How can we describe the properties of this set?

Let´s denote this set as set C. For the membership function of this set, μ_C could hold two conditions:

μ_C (7) = 1,

with an increasing difference $|x\text{-}7|$ its values should decrease to zero.

Then some of the possibilities are displayed in Figure 2, Figure 3, and Figure 4.

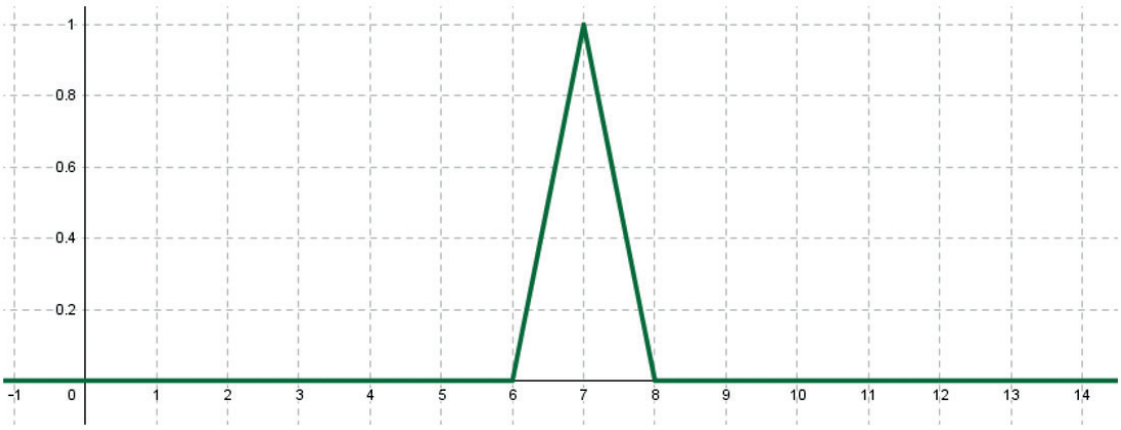


Figure 2: Triangular membership function representing the value "approximately 7"

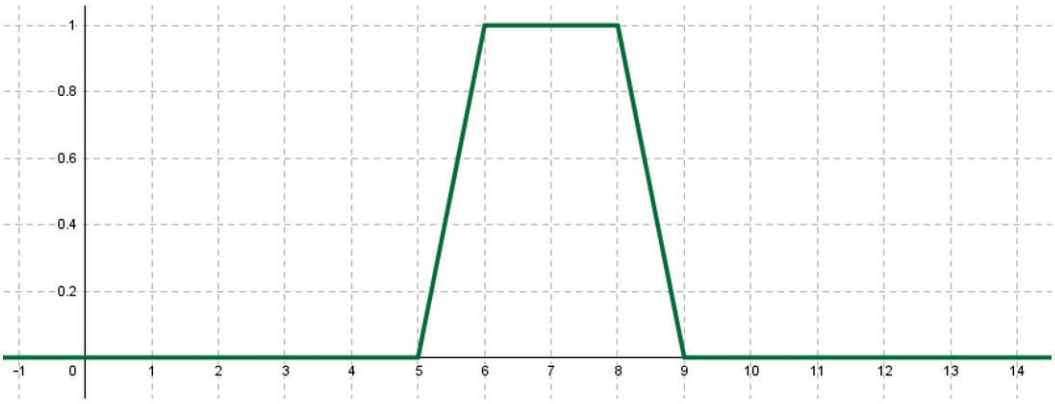


Figure 3: Trapezoidal membership function representing the value "approximately 7"

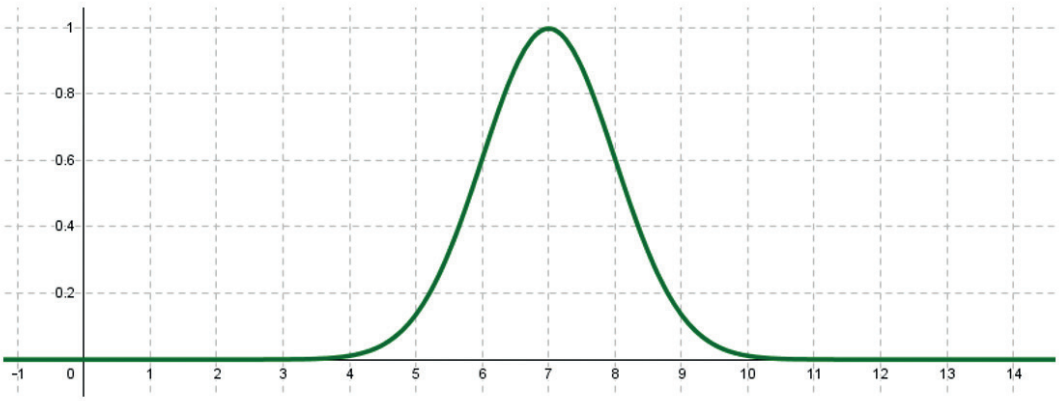


Figure 4: Another membership function representing the value "approximately 7"

The types of membership functions

In Figures 2-4, we saw that the membership function could have a lot of different forms. We will show some of them, which are defined in the software tool MATLAB.

Linear membership functions

Linear membership functions represent the simplest kind of membership functions. These are built with the use of parts of straight lines. They are divided into two basic groups:

- triangular,
- trapezoidal.

Triangular membership function

The triangular membership function consists of four parts (see Figure 5). The first part assigns an output value equal to zero (interval (-∞ *a*) on Figure 5) to input values. The second part is linearly increasing from value 0 to value 1 (interval (*a; b*) in Figure 5). Third part is linearly decreasing from value 1 to value 0 (interval(*b, c*) in Figure 5). The last part again assigns an output value equal to 0 (interval (c, ∞) in Figure 5) to input values. This membership function is generally described by three parameters *a, b, c*. In the software MATLAB it is denoted as **trimf,** and for parameters, we use the notation ***[a b c].*** Notice that the triangular membership function reaches an output value equal to 1 for only one input (specifically for input value *b*).

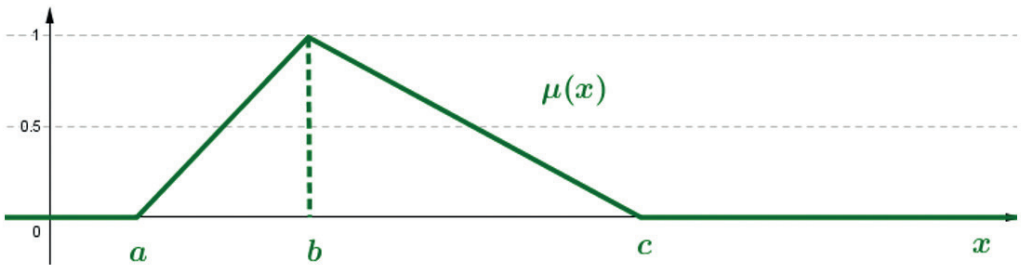


Figure 5: General triangular membership function

Trapezoidal membership function

The trapezoidal membership function consists of five parts (see Figure 6). In contrast with the triangular membership function, this function consists of the interval of input values, which reach the output value equal to 1. This membership function is generally described by four parameters *a, b, c, d*. In the software MATLAB, it is denoted as **trapmf,** and for parameters, we use the notation **[a b c d].**

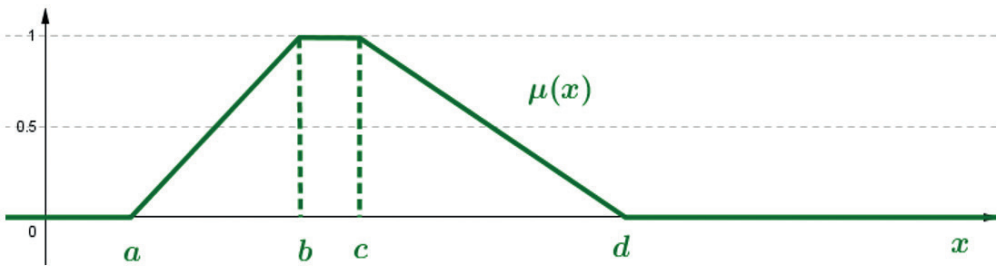


Figure 6: General trapezoidal membership function

Example: Write the MATLAB notation of the fuzzy sets A, B which are displayed in Figure 7:

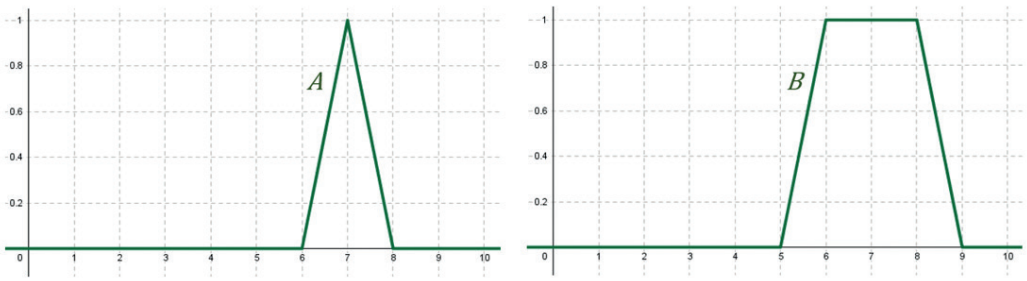


Figure 7: Fuzzy sets A and B from the example

Fuzzy set A is represented with the use of a triangular membership function. Its notation in the MATLAB program is *A [6 7 8]*. Fuzzy set B is represented with the use of a trapezoidal membership function. Its notation in the program MATLAB is *B [5 6 8 9]*.

Example: In real-life applications for some observed phenomenon, we often use the terms low, middle, and high phenomenon. For the temperature of the water, we could speak about low temperature, middle temperature, and high temperature (see Figure 8). While the term middle temperature (function M) could be described by a trapezoidal membership function, as mentioned in the previous text, the values low temperature (function L) and high temperature (function H) are specific in that sense that they need to be described with the use of asymmetric membership functions. How can we write the prescription of these functions in MATLAB software?

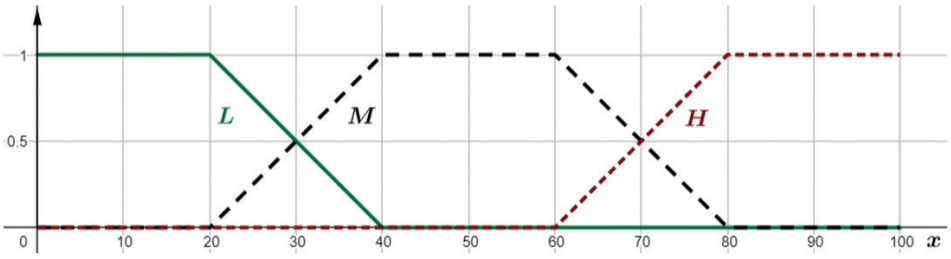


Figure 8: Fuzzy sets L, M and H from the example

To describe some fuzzy set in the MATLAB software, we could also use parameters that do not belong to the universe of the studied variable. So, in the first step we need to determine the universe of the term "water temperature". It is $\mathbb{X} = \langle 0, 100 \rangle$. Then we could write

$$L\ [-20\ -10\ 20\ 40], \quad M\ [20\ 40\ 60\ 80], \quad H\ [60\ 80\ 110\ 120].$$

Polynomial-based membership functions

This type of function is built by using the polynomial (quadratic) functions. They are divided into three basic groups:

- Pi curve,
- S curve,
- Z curve.

Membership function of Pi type

The membership function of Pi type is defined by six parameters a,b,c,d,e,f (see Figure 9). There are four parts of it, which are defined by quadratic functions (intervals $\langle a,b\rangle$; $\langle b,c\rangle$; $\langle d,e\rangle$; $\langle e,f\rangle$),), two parts where the value 0 (intervals $(-\infty,a\rangle$; (f,∞)) is assigned to each input value and one part where the value 1 (interval $\langle c,d\rangle$) is assigned to each input value. In MATLAB software, it is denoted as **pimf**.

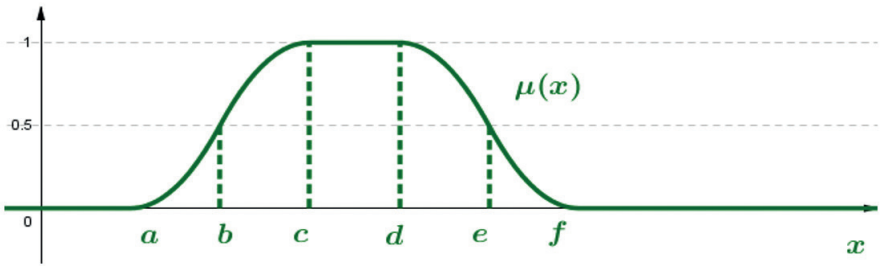


Figure 9: Membership function of type Pi

Membership function of type S

The membership function of type S is defined by three parameters a,b,c (see Figure 10). There are two parts of it which are defined by quadratic functions (intervals $\langle a,b\rangle$; $\langle b,c\rangle$), one part where the value 0 (intervals $(-\infty,a\rangle$) is assigned to each input value and one part where the value 1 (interval $\langle c,\infty)$) is assigned to each input value. In MATLAB software, it is denoted as **smf**.

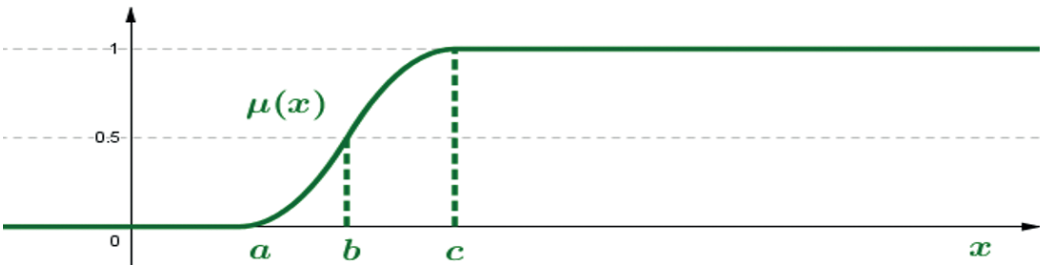


Figure 10: Membership function of type S

Membership function of type Z

The membership function of type Z is defined by three parameters a,b,c (see Figure 11). There are two parts of it, which are defined by quadratic functions (intervals $\langle a,b\rangle$; $\langle b,c\rangle$), one part where the value 1 (intervals $(-\infty,a\rangle$) is assigned to each input value and one part where the value 0 (interval $\langle c,\infty)$) is assigned to each input value. In MATLAB software, it is denoted as **zmf**.

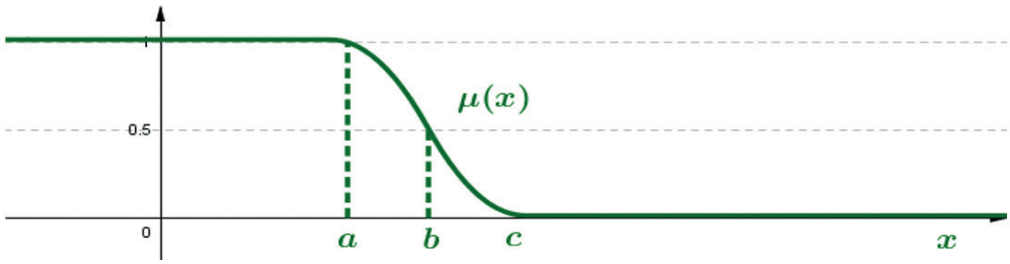


Figure 11: Membership function of type Z

Remark:

Membership functions of type S and Z represent asymmetric membership functions. They could be used for modelling of low and high values of variables.

Function-based on a statistical basis

If we have a large data set, we can process it using a statistical approach. **Gaussian membership functions (gaussmf)** is derived from the classical Gaussian distribution curve, which has two parameters c,σ (see Figure 12), where c represents the mean and σ represents the standard deviation of the data.

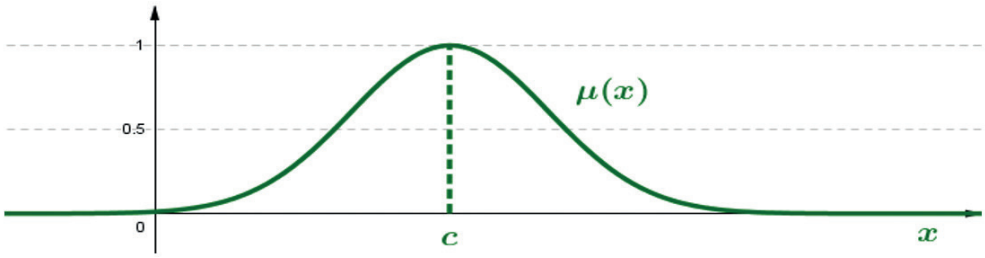


Figure 12: Gaussian membership functions

Sigmoidal membership functions

Gaussian membership functions are unable to **specify asymmetric membership functions**. For this reason, the **sigmoidal membership functions (sigmf)** with two parameters a, c are used (see Figure 13 and Figure 14). Then the parameters *a, c* are again obtained by using a statistical approach.

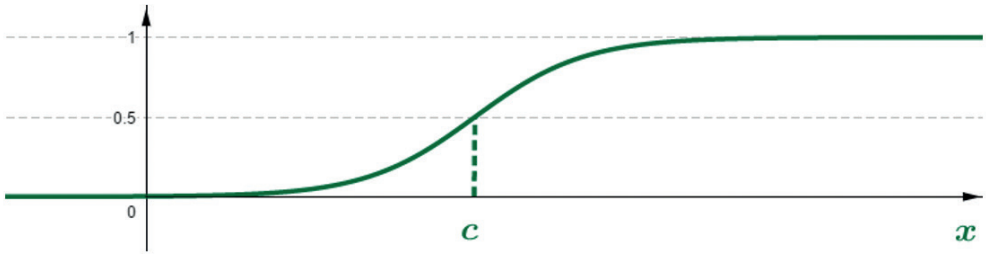


Figure 13: Sigmoidal membership functions where a > 0

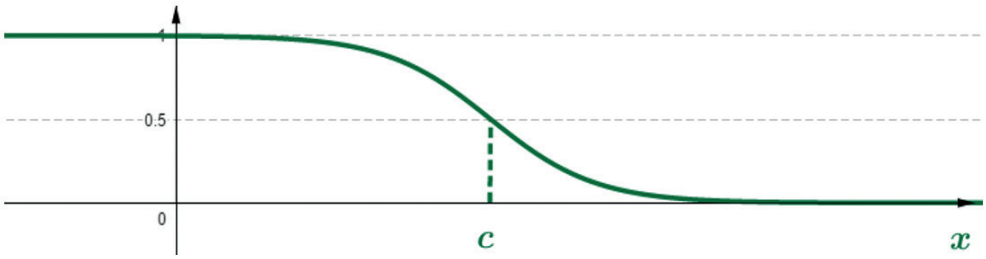


Figure 14: Sigmoidal membership functions where a < 0

Remarks:

We will use the **triangular** and **trapezoidal membership functions**. In real-life applications, Gaussian and sigmoidal membership functions are also often used, and their parameters are chosen using statistical data analysis.

In real life, we usually **first ask the expert** to describe the problem by suitable functions. Then, **in the second step**, we **usually specify the parameters of the functions** using (statistical) treatment of the big data group.

To work with fuzzy sets, we need to define the basic operations on fuzzy sets – **intersection, union,** and **complement**. Similarly, as many types of membership functions exist, **several types of operations on fuzzy sets are also defined**. We will mention so-called **standard operations on fuzzy sets**, which Professor Zadeh proposed.

Definition (standard intersection)

Let $\mathbb{X}$ be the universe and A, B be the fuzzy sets. **The standard intersection of two fuzzy sets** A, B is the fuzzy set $A \cap B$ with the membership function

$$\mu_{A\cap B}(x) = \min\big(\mu_A(x), \mu_B(x)\big).$$

Definition (standard union)

Let $\mathbb{X}$ be the universe and A, B be the fuzzy sets. The standard union of two fuzzy sets A, B is the fuzzy set $A \cup B$ with the membership function

$$\mu_{A\cup B}(x) = \max\big(\mu_A(x), \mu_B(x)\big).$$

Definition (standard complement)

Let $\mathbb{X}$ be the universe and A be the fuzzy set. **Standard complement of the fuzzy set** A is the fuzzy set $\overline{A}$ with the membership function

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

Example: There are two fuzzy sets A, B presented on the Figure 15. By using previous definitions graphically determine the intersection and union of fuzzy sets A, B and also complement to the fuzzy set A.

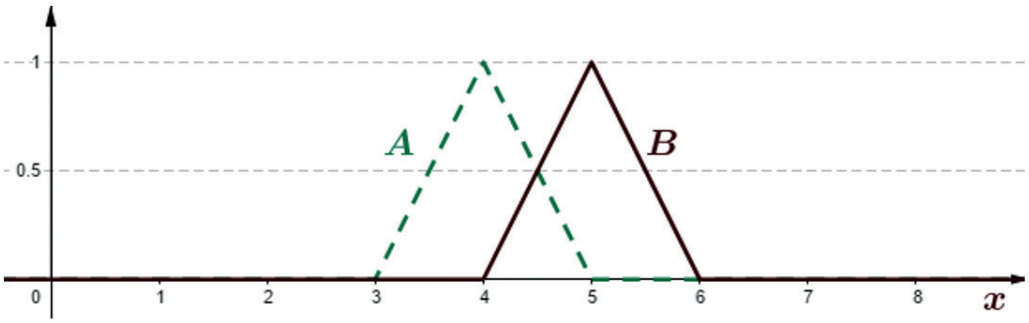


Figure 15: Fuzzy sets A, B from the example

The standard intersection of two fuzzy sets A, B is the fuzzy set $A \cap B$ with the membership function $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$. The solution is displayed in Figure 16.

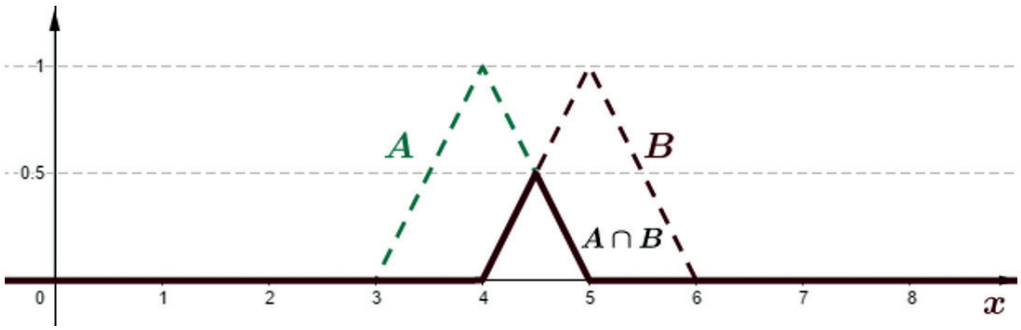


Figure 16: Standard intersection of fuzzy sets A, B from example

The standard union of two fuzzy sets A, B is the fuzzy set $A \cup B$ with the membership function $\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$. The solution is displayed in Figure 17.

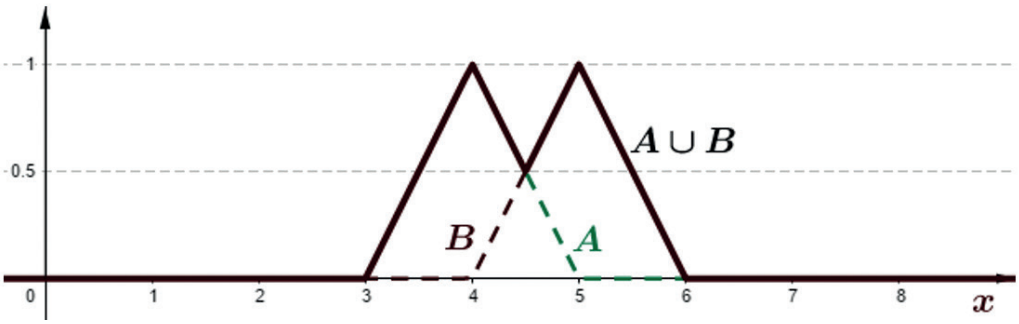


Figure 17: Standard union of fuzzy sets A, B from example

CHAPTER 6

FUZZY REASONING

This part of the handbook was written by Alžbeta Michalíková from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

Fuzzy reasoning is a process in which we deduce the consequents based on information that consists of vague terms. For example, in real life, we often use rules like

"If it is cold outside, I will dress in warm clothes."

We get these rules through observation, learning, reasoning, and so on. In fuzzy reasoning, we use the so-called IF - THEN fuzzy rules, which have the following form:

$$\underbrace{IF\ \langle \dots \rangle}_{= \textit{antecedent}\ (\textit{premise})} \qquad \underbrace{THEN \langle \dots \rangle}_{= \textit{consequent}\ (\textit{conclusion})}$$

For our needs, we will modify the rule

"If it is cold outside, I will dress in warm clothes."

into the form:

"IF the Temperature outside is low, THEN the Dress is warm."

The words **"Temperature"** and **"Dress"** are called **linguistic variables**. Therefore we write them capitalized. The values **"low"** and **"warm"** are called **the values of linguistic variables**. Linguistic variables which are in the antecedent part are called **input linguistic variables**. Linguistic variables, which are the consequence, are called **output linguistic variables**. By using the operations of conjunction (AND), disjunction (OR), and negation (NOT) we can create more complex rules, for example:

"IF the Temperature outside is low AND the Cloudiness is high, THEN the Dress is warm."

If we define all necessary rules, we get the set of rules, which is called the **rule base**. There are various approaches to working with the rules of the rule base. We will discuss and use one of them – the **Sugeno method**.

Sugeno method

The authors of this method are **T. Takagi**, **M. Sugeno,** and **G. Kang** [5]. They suggested it in the year 1985. This method was designed for the modelling of problems in which it is possible to describe the dependency between input and output variables by the function, which is non-linear, but there are some parts that are linear.

The Sugeno method was used for the first time in the **car parking** modelling problem. Today it is used for the **approximation of the data** by nonlinear functions in **classification, regulation** and **control, fuzzy decision-making, expert systems,** ...

In the Sugeno method, the values of **input variables** are described by **membership functions**. The expert designs them. The **output variables** are **described by functions** that could be either constant, linear, or **polynomial functions of any degree**.

Sugeno rules with constant output functions - a constant function describes the output variable of each rule. Generally, the rule has the form

$$R_j:\ IF\ X_1\ is\ A_{1j}\ \ AND\ \ X_2\ is\ A_{2j}\ \ AND\ \ \dots\ AND\ \ X_n\ is\ A_{nj}, THEN\ Y\ is\ b_j.$$

Sugeno rules with the linear output functions - the output variable of each rule is described by linear function. Generally, the rule has the form

$$R_j:\ IF\ X_1\ is\ A_{1j}\ \ AND\ \ \dots\ AND\ \ X_n\ is\ A_{nj},\qquad THEN\ Y\ is\ a_{1j}x_1 + \cdots + a_{nj}x_n + b_j,$$

where $a_{1j}, \dots, a_{nj}, b_j$ are real numbers.

Sugeno rules with the polynomial output functions - the output variable of each rule is described by a polynomial function of any degree.

$$R_j:\ IF\ X_1\ is\ A_{1j}\ \ AND\ \ \dots\ AND\ \ X_n\ is\ A_{nj}, THEN\ Y\ is\ a_{1j}{x_1}^{m_1} + \cdots + a_{nj}{x_n}^{m_n} + b_j,$$

where $a_{1j}, \dots, a_{nj}, b_j$ are real numbers and $m_1, \dots, m_n$ are natural numbers.

Example: Sugeno rule with the constant output function

We had to evaluate the students with the use of the Sugeno method. It could be described by the rules of type

R: IF Presentation is high AND Test points value is high,

THEN Evaluation is equal to 1 (=A).

Example: Sugeno rules with the linear output functions

We know that for some values (for example, for small values) of the car's position, the car will go along a straight-line prescription which can be easily determined. The rule has the following form

R: IF Position value is low,THEN Line prescription is 3.25x+2.5 .

Let us have the rule base with the k rules. Let there be $\mathbb{x} = (x_1, x_2, \dots, x_n)$. Let each rule have an output function in the form $y_j = a_{1j}{x_1}^{m_1} + a_{2j}{x_2}^{m_2} + \cdots + a_{nj}{x_n}^{m_n} + b_j$. Then the final output y_x is calculated by the formula

$$y_{\mathbb{x}} = \frac{\sum_{j=1}^{k} w_j y_j}{\sum_{j=1}^{k} w_j}$$

where w_j is the weight of the j-th rule (see Figure 18).

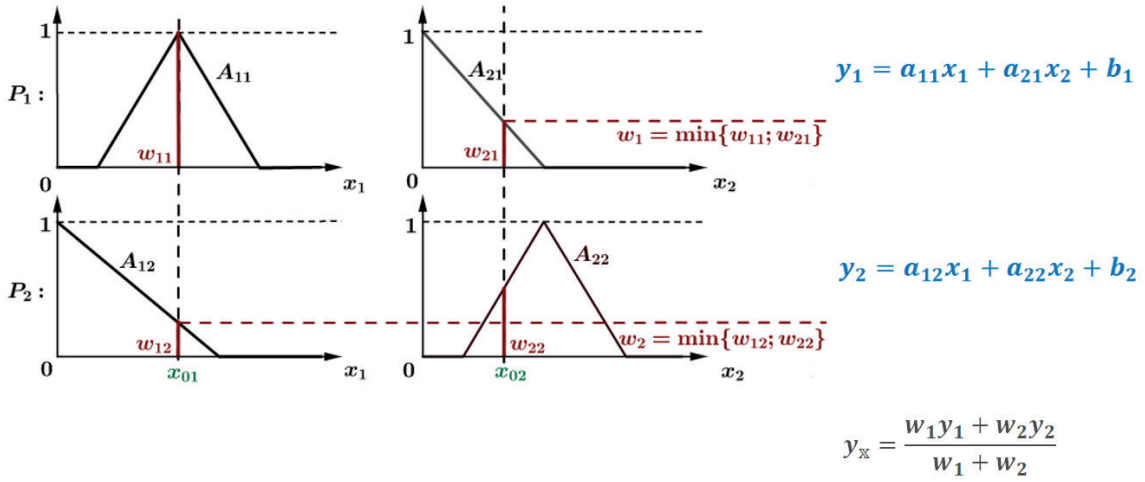


Figure 18: Final output in the Sugeno method with two input variables and two rules

Remark on the way of obtaining weights:

Let us have rule R_j with n input variables $(x_1, x_2, \dots, x_n)$. Firstly, we compute weights w_{ij} as the intersection between the value of measured input x_i and the respective membership unction A_{ij}. Secondly, we compute the weight w_j by using the following formula

$$w_j = \min_i w_{ij} .$$

How to appropriately design fuzzy rules?

- We can ask experts to describe their knowledge using the appropriate functions.
- We can determine the parameter values of the functions by processing a large amount of known data.

This method has several names, for example, the **Sugeno method, Takagi-Sugeno fuzzy inference system, Takagi-Sugeno regulator**, ... These names represent the same method. The name used is related to the area in which the method is used.

We will demonstrate the use of the Sugeno method in two different areas

- in data classification,
- in data approximation.

In both cases, we will be the experts who will design the rules of the given system [2], [6], [7]. To create the values of **input linguistic variables**, we will use the simplest types of membership functions - **linear membership functions**. To create the values of the **output linguistic variables**, we will use **constant functions for classification** and **linear functions for approximation**.

CHAPTER 7

USING SUGENO METHOD FOR DATA CLASSIFICATION

This part of the handbook was written by Alžbeta Michalíková from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

In this part of the handbook, we will work with the **Iris dataset** (see Appendix A). Recall that the Iris dataset consists of 150 samples of Iris flowers. For each flower, we have **four basic attributes** - the length and the width of the sepals and the length and the width of the petals in centimeters (or millimeters). On the other side, we could classify the flowers into three classes corresponding to three species of Iris (**Iris Setosa, Iris Virginica,** and **Iris Versicolor**).

In this part, we will combine the data processing in software Excel and MATLAB.

Example: Classify data from the Iris dataset into a suitable number of classes using the Sugeno method. (Solution of this example can be found in Appendix B.)

First, let´s try to answer the following questions:

1. *How many **input variables** are there in the Iris dataset?*
2. *What will we use **to describe input variables**?*
3. *What **type of fuzzy membership functions** will we use?*
4. *What will be the **output**?*
5. *What will we use **to describe the output variables**?*
6. ***Which type of rules** will we use?*
7. ***Write an example** of one rule!*

Second, download the Iris dataset from some webpage and copy it into the Excel file. Let's **mark** the first 50 entities with **red color**, the next 50 entities with **blue color,** and the rest with **green color**. In Excel, create four independent sheets and copy the colored table into each of them. In the first sheet, sort the values (from smallest to largest) according to the first column. Similarly, sort the values in each sheet according to one of the columns. To model the input variables, we will use **trapezoidal functions**. Determine the values of input variable parameters from these data and fill them into the following tables.

INPUT 1:

Name	Parameters
Universe	
Red	
Blue	
Green	

INPUT 2:

Name	Parameters
Universe	
Red	
Blue	
Green	

INPUT 3:

Name	Parameters
Universe	
Red	
Blue	
Green	

INPUT 4:

Name	Parameters
Universe	
Red	
Blue	
Green	

Table1: Parameters of input variables

Third, determine the values of output parameters. Fill the following table with the correct values if, for the **output linguistic variable**, we use **constant functions**.

OUTPUT:

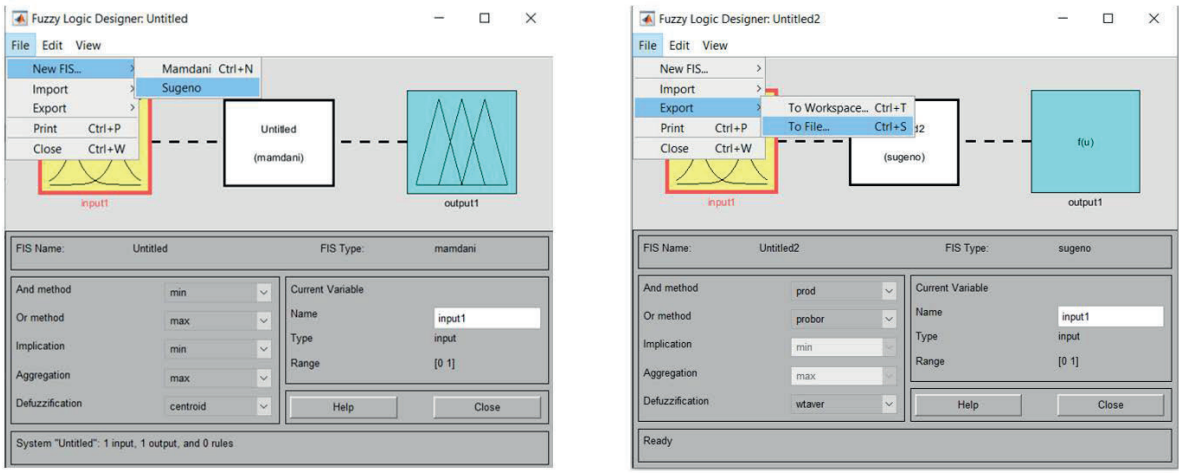
Name	Parameters
Universe	
Red	
Blue	
Green	

Table 2: Parameters of the output variable

Fourth, suggest the number of rules and write them correctly.

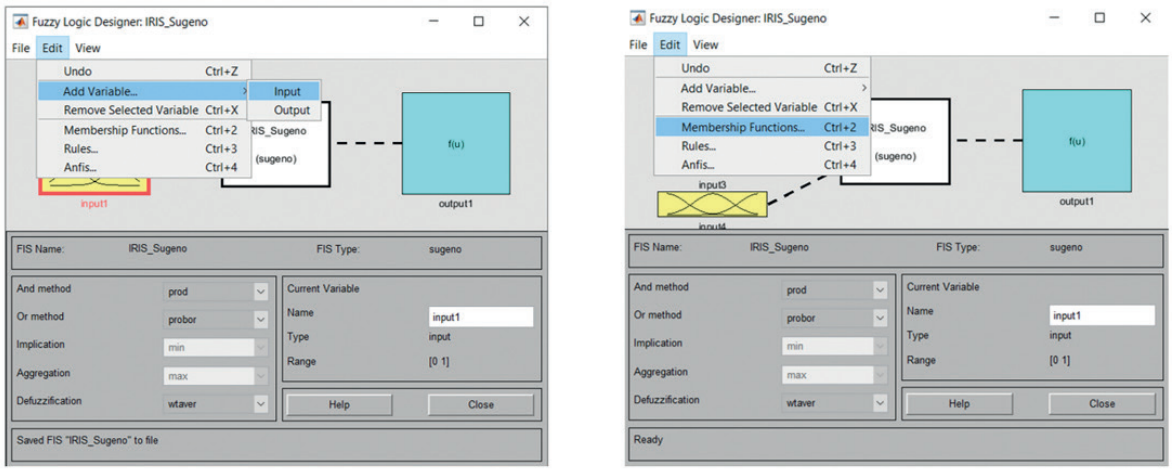
Rules:

Now we will process the obtained values in MATLAB software. Open MATLAB software and write the command **fuzzy** into the Command Window. This command opens the Fuzzy Logic Designer. We will use the Sugeno method (Sugeno fuzzy inference system = Sugeno FIS). Therefore, we need to open this type of FIS (see Figure 19a). We could rename and save this FIS, for example, as a file **IRIS_Sugeno** (see Figure 19b). Now we need to have four input linguistic variables – we add three new input variables (see Figure 20a), and then we need to edit the parameters of membership functions (see Figure 20b). Now for each input variable, we will change the range of the variable step by step, add the name of the membership functions, change the type of membership functions, and add the parameters to each membership function (use Table 1). These steps are shown in Figure 21.



(a) (b)

Figure 19: Opening new Sugeno FIS (a) and renaming/saving FIS (b) in software MATLAB



(a) (b)

Figure 20: Adding new variables (a) and editing membership functions (b) in MATLAB software

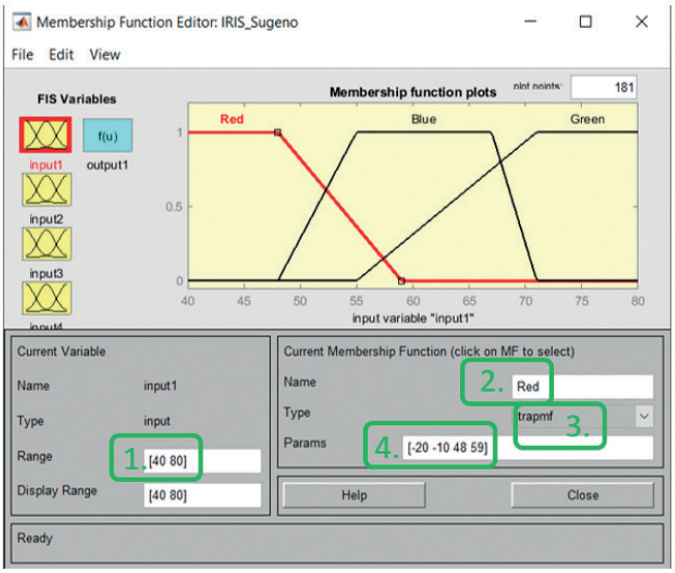


Figure 21: Changing the input membership function parameters in MATLAB software

Now we need to change the values of the output variable. To edit the output variable, we use a double-click on the blue rectangle named output1. We will get the new menu for the output variable, as presented in Figure 22. We will fill the values from Table 2 into this menu.

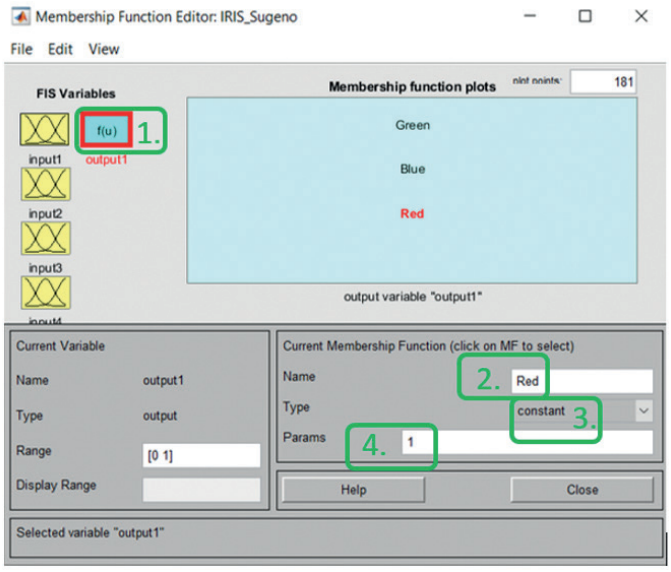
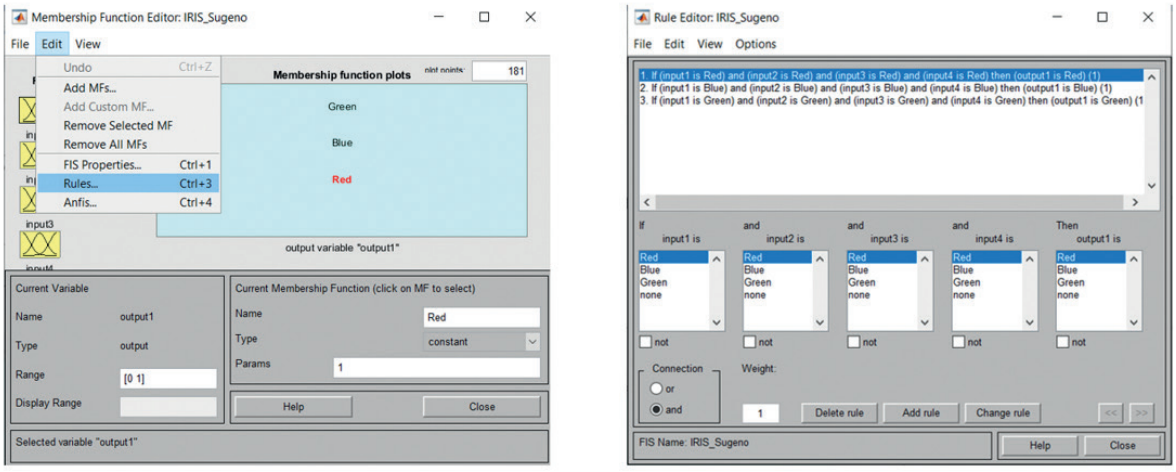


Figure 22: Changing of output values parameters in MATLAB software

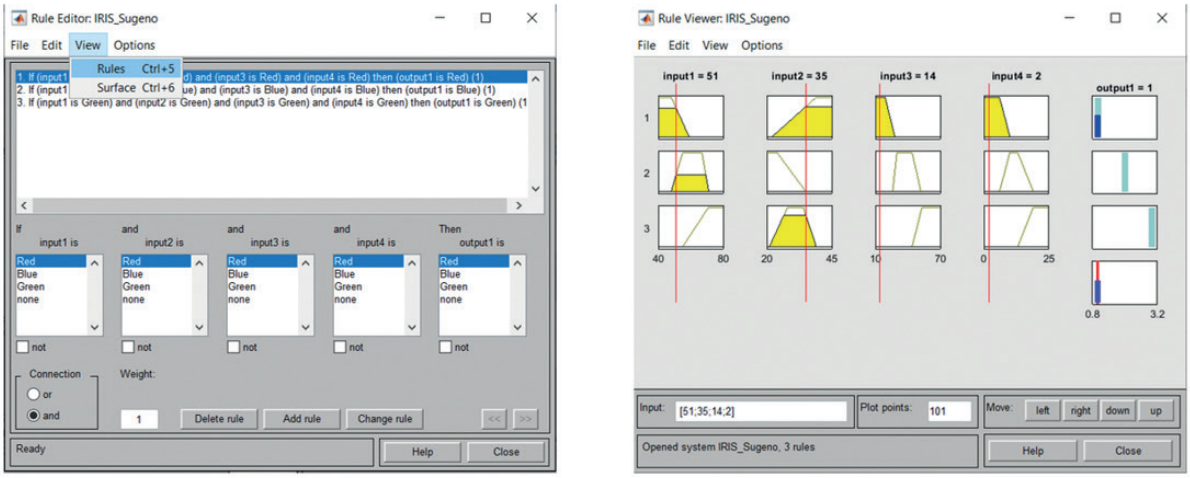
Our last step is to create the rules of our system. We open the rule menu (see Figure 23a) and use three simple rules (see Figure 23b).



(a) (b)

Figure 23: Opening the rule menu (a) and adding the rules (b) in MATLAB software

Our system is ready. Now we can evaluate the results that the system gives for known inputs. We can open the rule viewer (see Figure 24a) and add a specific value to each input variable (see Figure 24b). We can add these values by moving red lines on the top of the menu or by changing the values of the parameters of the orderly foursome on the bottom of the menu.



(a) (b)

Figure 24: Opening the rule viewer (a) and adding specific values to input parameters (b) in MATLAB software

Remarks:

The input values displayed in Figure 24b belong to the first row of the Iris data table. As we can see, the system classified the object with these input attributes into class 1 (output1 = 1). It is the value that we expected as a result!

We have shown how we can classify one object from the Iris dataset. We can use this approach for each row of the table. Of course, we can also classify all table rows in one step by using a sequence of commands from the command line. We can also calculate the success rate of the classification using the given FIS. Using the parameters mentioned in Appendix B, we reached a success rate of 94.6667%, i.e., 142 flowers from all 150 were classified correctly.

The success rate of the classification can be improved with the use of several different approaches. For example, we can use more values of input linguistic variables and then create more rules. We used three values for each input variable (**red – blue – green**). We could also use five values of each input variable, which represent the values **very_small_value – small_value – middle_value – high_value – very_high_value**. Then we can create more rules by combining the values of these input variables. On the other hand, we can use other methods, which were designed to optimize the parameters of values of input and also output variables. One of them is the so-called **ANFIS = Adaptive Neuro-Fuzzy Inference System**, which optimizes the parameters of created FIS using a Neural Network. The basic knowledge concerning neural networks is presented in the next part of this handbook.

CHAPTER 8

USING SUGENO METHOD FOR DATA APPROXIMATION

This part of the handbook was written by Alžbeta Michalíková from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

In this case, we have a lot of data and need to process it. It is often useful to approximate these data by some simpler function that gives an approximation of real output for exact input values. This process is called **approximation**. Sugeno method was designed for approximating such data on some parts of the domain linear (in 2D they can be approximated by part of the line). On the rest of the domain, they must be approximated by some appropriate function. We will approximate data representing the car's path in this part of the text.

In this part, we will combine the data processing in software Excel and MATLAB.

Example: Let's imagine that we are developing an autonomous vehicle. One of the problems which we need to solve is to find the function which will describe the car parking into some parking place. We could ask the professional driver to park at some specific place number of times, and we could capture the car's path by sensors (see Figure 25).

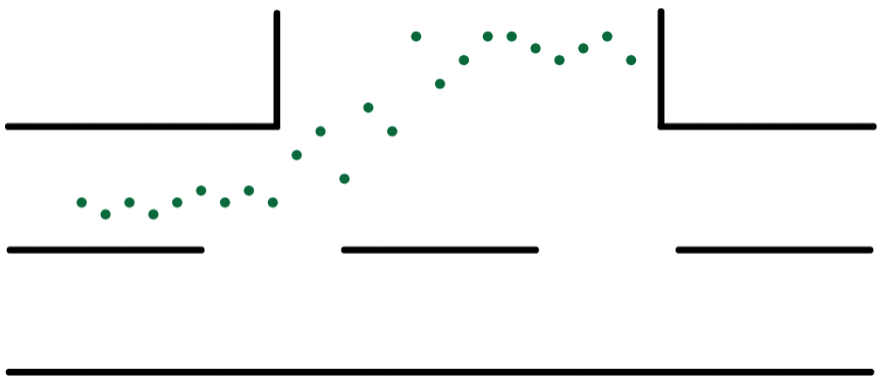


Figure 25: Position of the car during the car parking process

Solution: This car movement can be described by lines in the two parts. The first part - movement on the straight road before parking. Second part - movement on the parking place. The movement between these two parts will be approximated by the Sugeno method.

In the first step, we will place our data into the Cartesian coordinate system (see Table 3 and Figure 26). We can also draw the lines of straight movement and name them y_1 and y_2. As we can see, some data points will contribute to the description of one line (points with their x value from intervals (1,10) and (15,24) and also data that will contribute to the description of two lines (points with their x value from intervals (10,15). This information is important when we design the membership functions of the used fuzzy set.

Table 3: Coordinates of approximated data

x	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
y	*8*	*7,5*	*8*	*7,5*	*8*	*8,5*	*8*	*8,5*	*8*	*10*	*11*	*9*
x	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*
y	*12*	*11*	*15*	*13*	*14*	*15*	*15*	*15*	*14*	*15*	*15*	*14*

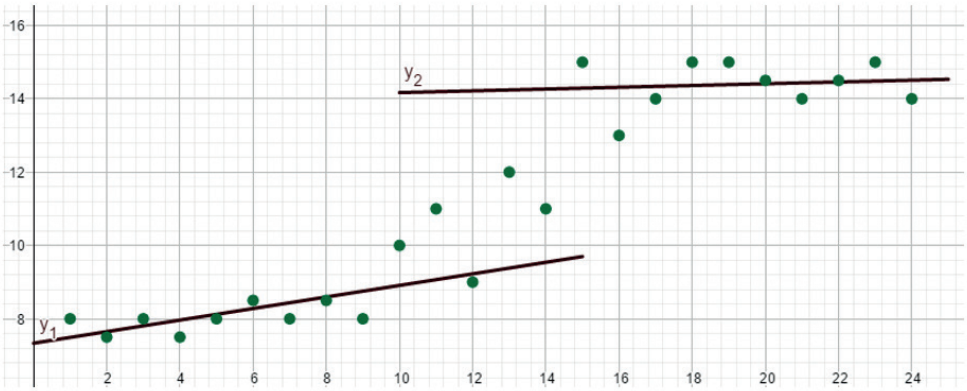
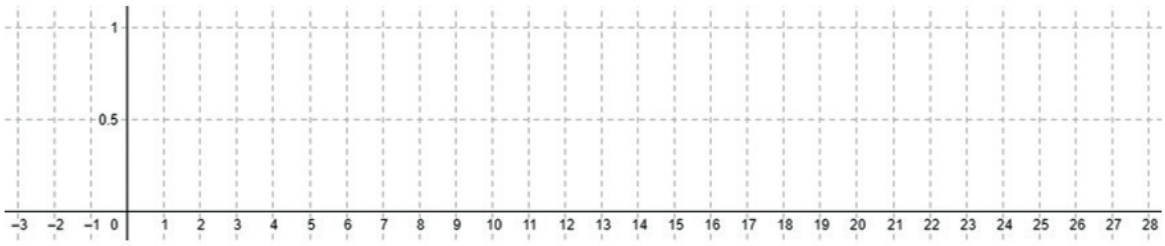


Figure 26: Place data into the Cartesian coordinate system

Let´s answer the following questions:

1. How many **input variables** do we have? Name these variables!
2. How many **values of input variables** will we use? Name these values of variables!
3. What will we use **to describe input variables**?
4. What **type of fuzzy membership functions** will we use?
5. Can you **draw** these **membership functions**? Use the following grid:



6. Can you **write the universe** and **parameters** of these functions? Can **you write the prescription** for these functions for MATLAB software?
7. What will be the **output**?
8. What will we use **to describe the output variables**?
9. **How many rules** will we use?
10. **Write an example** of one rule!

Answers:

There is just one input variable – it could be named **Position** (of the car) **on the *x*-axis**. This input variable has two values – A **Low value of coordinate *x*** and a **High value of coordinate *x***. We will describe them by fuzzy sets. We will use **trapezoidal membership functions**. They could be drawn as it is displayed in Figure 27.

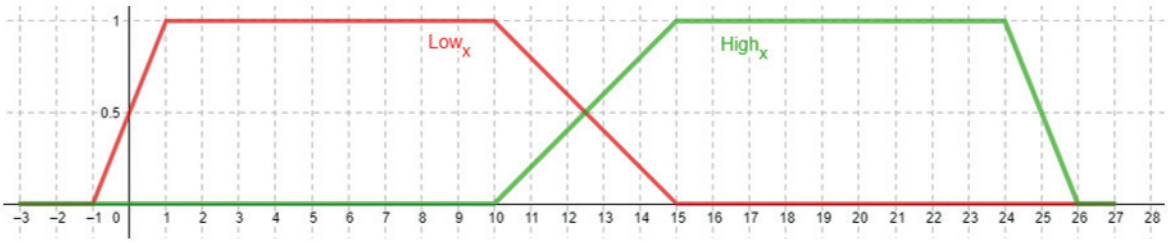


Figure 27: Trapezoidal membership functions for data approximation

The universe of these fuzzy functions is X=[1,24]. For the Low value of coordinate x we have the parameters *Low x*=[-1,1,10,15]. For the High value of coordinate x we have the parameters *High x*=[10,15,24,26].

The output variable represents the **Position of the car on *y*-axis**. As an output, we will use a linear function (line). We will use two lines y_1 and y_2. To describe the parameters of these lines, **we will use software Excel** (see below). Then we will have **two IF-THEN rules** which could be written as follows:

R1: IF Position of the car on x axis is Low_x, THEN Position of the car on *y* axis is y_1.

R2: IF Position of the car on x axis is High_x, THEN Position of the car on *y* axis is y_2.

We need to compute the parameters of linear functions, which represent the output of the rules. These parameters will be computed from those data values, which contribute to description of exactly one line, i.e., to the description of line y_1 contributed first 10 data points. Let´s copy them to Excel –

each point into one row (see Figure 28a). Then mark these points and use the following steps: **Insert** → **Charts** → **Points**.

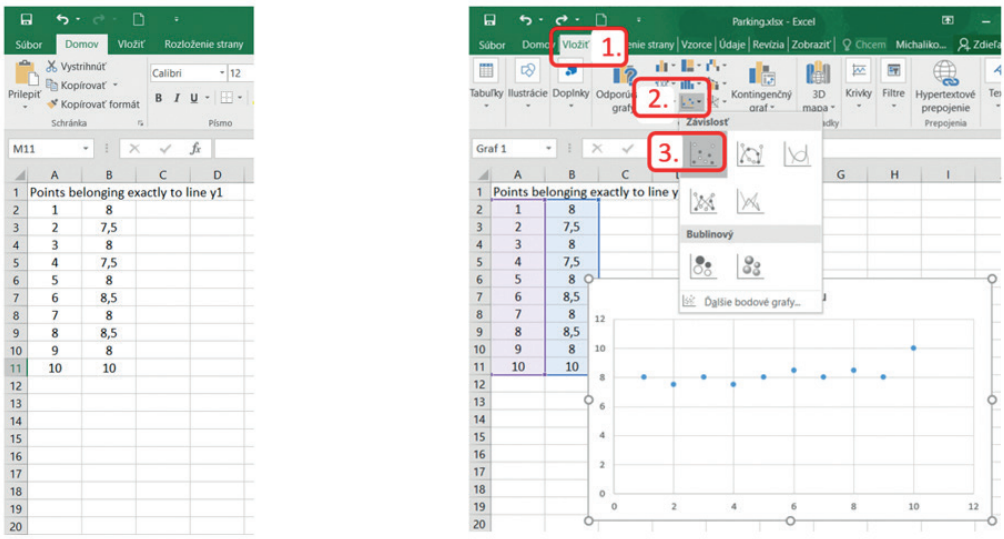


Figure 28: Processing input data in software Excel

Then add the element of the graph as shown in Figure 29a and display the equation of the line (see Figure 29b).

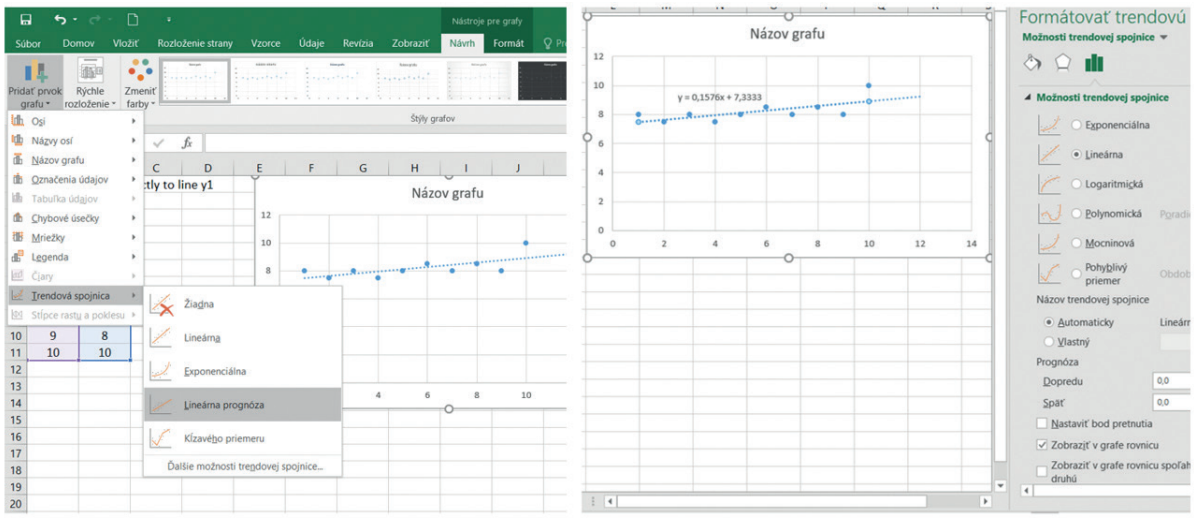


Figure 29: Displaying the equation of the line in Excel software

We got the parameters of line y_1. Using the same procedure, we will get the parameters of line y_2. Then y_1=0,1576x + 7,3333 and y_2= 0,0242x + 13,927. In MATLAB prescription, it is y_1 [0,1576 7,3333] and y_2 [0,0242 13,927].

Table 4: Summary of the input and output parameters for data approximation

INPUT:

Name	Parameters
Universe	[1, 24]
Low x	[-1, 1, 10, 15]
High x	[10, 15, 24, 26]

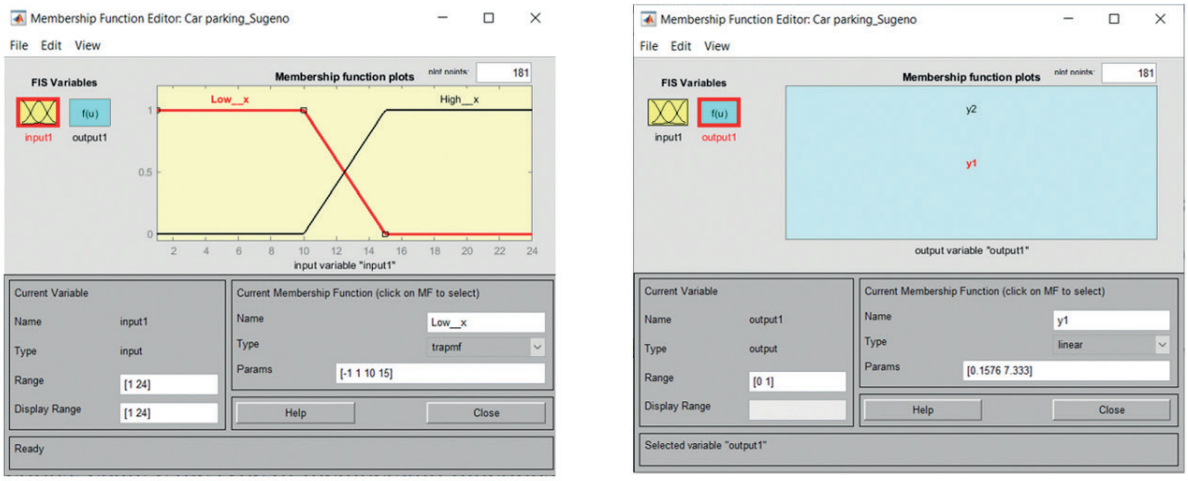
OUTPUT:

Name	Parameters
Universe	---
y_1	[0,1576 7,3333]
y_2	[0,0242 13,927]

Now we have all the parameters, and we can create the FIS of the Sugeno type in MATLAB software. The first steps are similar to those mentioned in subsection 3 (data classification). Open MATLAB software, and into the Command Window, write the command **fuzzy**. This command opens the Fuzzy Logic Designer. We will use the Sugeno method (Sugeno fuzzy inference system = Sugeno FIS). Therefore, we need to open this type of FIS (see Figure 19a). For example, we could rename and save this FIS (see Figure 19b) as a file **Parking_Sugeno**.

We have just **one input linguistic variable**. For this variable, we have just **two membership functions**. To remove one of them, click on one of the functions on the graph and use Delete on the keyboard. Then edit the parameters of membership functions (use values from Table 4). The final configuration for input membership functions is displayed in Figure 30a.

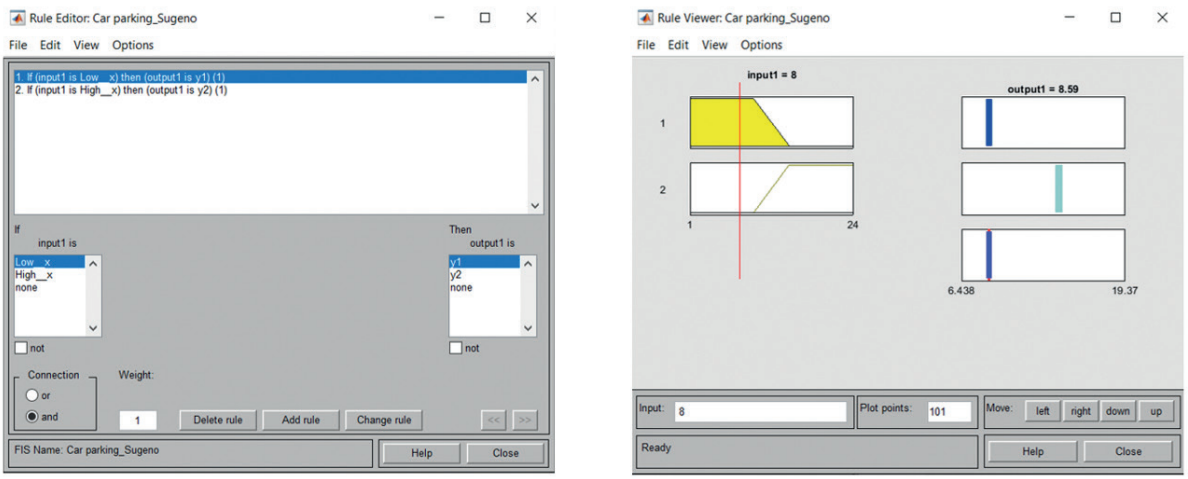
Now we need to change the values of the output variable. To edit the output variable, we again use double-click on the blue rectangle named output1. We will get the menu for the output variable. Similarly, as it was at the creation of the previous FIS, we fill all the values (from Table 4). Remember that in this FIS, the type of **output function is linear** (see Figure 30b).



(a) (b)

Figure 30: Configuration of input and output variables in MATLAB software

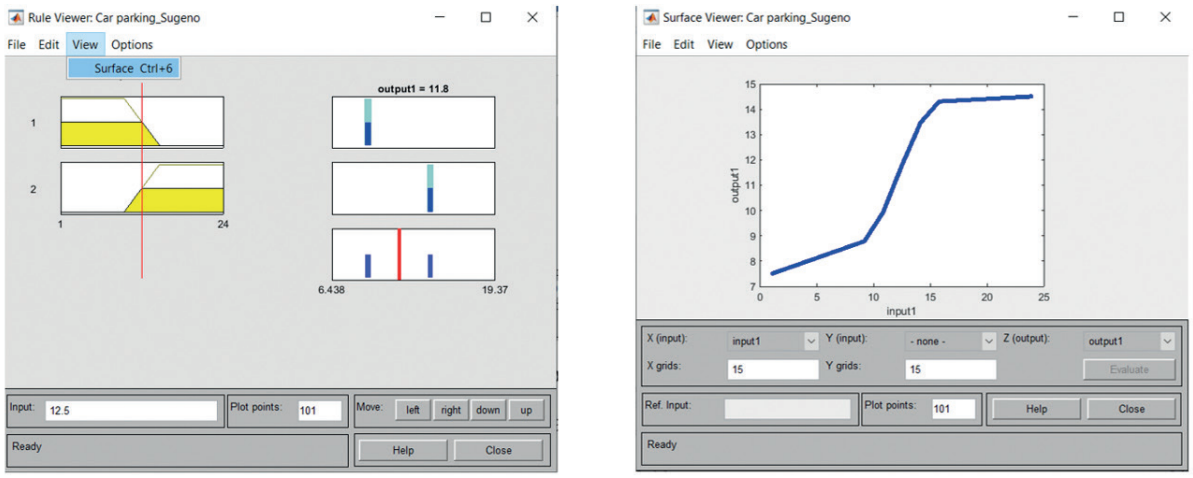
Our last step is to create the rules of our system. We will use two simple rules (see Figure 31a). Our system is ready. Now we can evaluate the results that the system gives for known inputs. We can open the rule viewer and add to the input variable its specific value (see Figure 31b).



(a) (b)

Figure 31: Configuration of rules and evaluation of the results in MATLAB software

There is also the possibility to show the function that we create using this FIS (see Figure 32a).



(a) (b)

Figure 32: Opening the Surface Viewer and final function of created FIS in MATLAB software

Remarks:

The input value displayed in Figure 31b is equal to 8. As we can see, the system gave an output of 8.59 to this input value. The real value (see Table 3) was 8.5. Therefore, the obtained value represents a good approximation for this point.

We can compare the original (real) data with the obtained function. There are two basic approaches to comparing (evaluate) the results. The first one is graphical. The second one is by computing error of the created system. Figure 33 shows a **graphical comparison** of real data and obtained function.

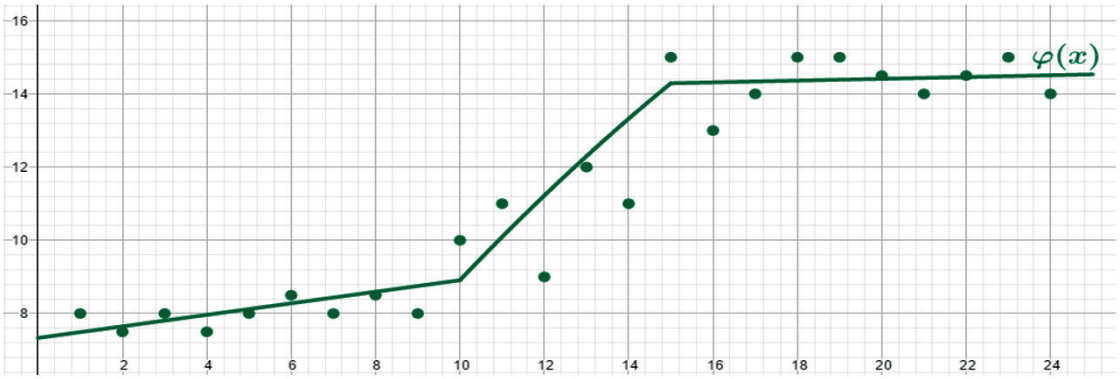


Figure 33: Graphical comparison of real data and obtained function

We have shown how to get approximated output value to one specific input value. Of course, we can approximate all the inputs of Table 3 in one step with the use of a sequence of commands from the command line. We can also calculate the obtained error. More types of errors could be computed. The so-called Mean Square Error (MSE) is the most widely used error. This error is computed by the formula,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[f(x_i) - \varphi(x_i)]^2 ,$$

where n represents the number of input data, values of $f(x_i)$ represent the real outputs, and $\varphi(x_i)$ represent the outputs computed by FIS. This MSE value is suitable to use if we want to compare two or more different approaches. Then the smallest value represents the better approach. For our system, we reached the value of MSE equal to 0,7263.

The quality of approximation can be improved with the use of several different approaches. For example, we can use more membership functions and then create more rules. In the presented example, we used two membership functions (**Low_x – High_x**). We can use three input variable values, representing the values **Low_x – Medium_x – High_x**. Then we can find the prescription of 3 lines and create three rules by combining the input and output values. When we have a large set of inputs, we can also use another type of membership function (they were mentioned in subsection 2 of this section). The statistical distribution of the data can then determine the parameters of the functions. On the other hand, we can again use another method, which was designed to optimize the input values parameters and output variables.

CHAPTER 9

INTRODUCTION TO OPTIMIZATION

This part of the handbook was written by Fatih Kilic from Adana Science and Technology University in Adana, Turkey.

In many areas of study, an optimization problem can be tackled to search for the optimal solution in search space (all feasible solutions) using mathematical, heuristic, and meta-heuristic methods. There are different optimization problems such as engineering, financial, medical, and production problems (Mirjalili, 2016, Cui et al., 2017, Kilic et al., 2021, Aktaş et al., 2022). Figure 1 shows the main steps of solving optimization problems. In the first step, decision-makers want to solve an optimization problem to improve current systems or suggest new systems. For example, we want to locate the hospital in the best position considering demand and potential patients. Secondly, this problem must be mathematically formulated as solution structure, objective, and constraints. The solution structure consists of decision variables. Decision variables are possible positions of candidate hospitals for this problem. The objective function measures the quality of a solution to evaluate among candidate solutions. The objective function can be the sum of distances between hospitals and potential patients for the example problem. All solutions might be feasible or unfeasible solutions because of predefined constraints. These constraints are defined by the specialist. For this problem, the least one hospital might be in a sub-area that is demanded by stakeholders. In the third-step, well-known methods are implemented to find good solutions. These methods generate an optimal solution or good solutions close to the optimal solution. Stakeholders interpret the solutions and make any minor revisions of the solution if it is necessary. Finally, the solution is realized.

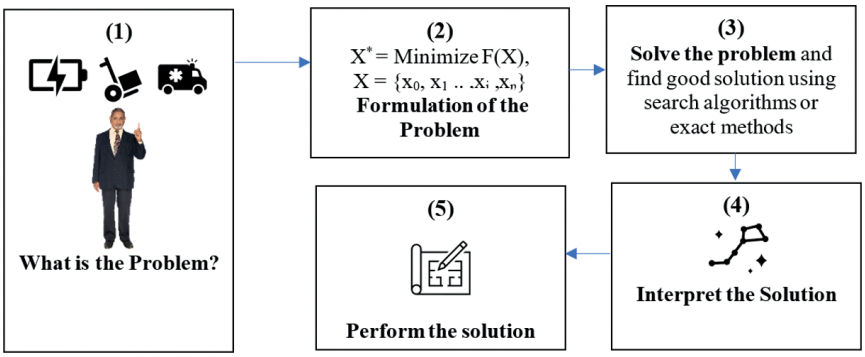


Figure 1: Main steps of the solving optimization problems

Mathematically definition, any optimization can be explained as follows:

$$\max/\min_{x \in F \subseteq S} f(x), \qquad \text{Eq. (1)}$$

where x shows a set of decision variables, F contains feasible solutions, S represents solutions space, and $f(x)$ demonstrates objective function. max/min aim to find the maximum and minimum values of $f(x)$.

We can formulate constraints and a range of data and variables. An example is given as follows:

$$\sum_{j}^{n} x_j < \mathrm{b} \qquad \text{Eq. (2)}$$

$$x_j \in \{0,1\} for\ j = 1 \ldots . n \qquad \text{Eq. (3)}$$

where x_j can be 0 or 1 for all j, and the sum of all x items less than b according to Egs. (2 and 3).

Problems trying to find continuous variables are classified as continuous optimization problems. Table 1 shows well-known continuous optimization problems. The solution (X) consists of D-dimensional real values. Each dimension is between the predefined minimum and maximum numbers. The dimension is number of decision variables. These are used to demonstrate their performance when optimization algorithms are introduced.

Table 1 - Unimodal Functions (Li et al., 2013, Hayyolalam, 2020, Wang et al., 2022).

Dimension	Range	Equation
5	[-100, 100]	$F_1(x) = \sum_{i=1}^{n} x_i^2$
	[-10, 10]	$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
	[-100, 100]	$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$
	[-100, 100]	$F_4(x) = \max_i \{ \lvert x_i \rvert,\ 1 \le i \le n \}$
	[-30, 30]	$F_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$
	[-100, 100]	$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$
	[-1.28, 1.28]	$F_7(x) = \sum_{i=1}^{n} i x_i^4 + \mathrm{random}\,[0, 1]$

9.1 Local Search Algorithms

Local search algorithms (LSAs) are used to solve optimization problems in computer science and related computational sciences (Kiliç & Gök, 2013, Hoos & Stützle, 2004). These algorithms try to solve the generic optimization of scalar functions (Rossi, 2006). However, they can be implemented for different optimization problems after formulating the optimization problems.

Commonly LSAs deal with a single solution to produce a better solution at any time. Simulated annealing (Bertsimas & Tsitsiklis, 1993), Tabu search (Glover & Laguna, 1998), Hill climbing, and variable neighborhood search (Hansen & Mladenovic, 1999) are well known local search algorithms.

Algorithm 1 shows the main steps of the hill climbing (HC) algorithm.

Algorithm 1: Hill Climbing	
1	**currentSolution** = Generate initial solution
2	**Evaluate currentSolution**
3	**iteration =0**
4	**while** (!Stop contitions)
5	**NeighbourSolution** = Movement(**currentSolution**)
6	**If** NeighbourSolution is better than **currentSolution**
7	**currentSolution = NeighbourSolution**
8	**end if**
9	**iteration = iteration+1**

In the first step, the initial solution is generated randomly, and it is assigned to the current solution. For example, X= [60.15, -50.07, 10.08, -80.01, 17.59] for F1 function in Table 1. Each item of this vector is between -100 and +100 and is randomly selected. Secondly, the neighbor solution is generated using the current solution and minor modification function in each iteration. Random index number is selected, and the selected item is changed for X vector. If the neighbor solution is better than the current solution then the current solution is updated with the neighbor solution. The iterations are performed until the current solution is satisfied or the maximum iteration.

9.2 Evolutionary Computation

Evolutionary computation (EC) is a popular optimization and population-based algorithm that mimics biological evolution such as reproduction, recombination, mutation, selection and survival of individuals (De La Iglesia, 2013, Bartz-Beielstein et al., 2014). Different variants of EC are introduced using biological evolution processes. Genetic algorithm is invented by John Holland (1962) while Evolution Strategies is invented by Ingo Rechenberg (1965).

The steps of a typical EC is given by Algorithm 2.

Algorithm 2: Evolutionary Computation	
1	**Population** = Generate randomize initial solutions
2	**iteration =0**
3	**while** (!Stop conditions)
4	Fitness values are computed for each individual in **Population**
5	Individuals are selected as parents in **Population** based on fitness values
6	Crossover and mutation are performed to generate offspring
7	Update **Population** according to new offspring and their fitness values
8	**iteration = iteration+1**

In the first steps, a population is created randomly as a predefined size. A Population consists of solutions. Each individual represents a solution. The second step is a set of iterative processes. In the second step, the fitness value is computed for each individual using an objective function. Parents are selected based on their fitness or different techniques. Crossover and mutation are reproduction processes to generate new solutions. For next-generation, the selection process is performed using the individuals' fitness values. These steps are repeated until stop conditions are satisfied.

Crossover Operator

Crossover operator is to exchange information among two selected parents' chromosomes to generate two new offspring (Kilic et al., 2021, Ahmed, 2010). This operator is an important exploitation operator in EC. There are different general crossover techniques such as Single-point, multi-point, uniform crossovers, and problem-specific crossover techniques (for combinatorial optimization problems) such as edge recombination, multi-parent partially mapped crossover, and order-based crossover. This operator is performed according to crossover probability.

Table 2 represents single-point crossover operator example. Parents 1 and 2 are selected individuals, showing two solutions as italic and underline, respectively in the first and second rows. A cut point is selected randomly, and parents are divided into two parts for each individual. Children 1 and 2 are generated to swap the second parts of parents and to take the same first parts of parents.

Table 2 - The example of a Single-Point crossover operator

	X_1	X_2	X_3	X_4	X_5
Parent 1	*60.15*	*-50.07*	*10.08*	*-80.01*	*17.59*
Parent 2	40.22	30.08	20.09	-20.05	60.85
Child 1	60.15	-50.07	20.09	-20.05	60.85
Child 2	40.22	30.08	*10.08*	*-80.01*	*17.59*

Mutation Operator

A mutation operator is performed to ensure diversity in the population. The mutation operator modifies a parent to produce offspring. A random position of the solution is selected and the corresponding gen or bit is changed to perform the mutation operator. There are different mutation operators. One of the mutation operators is Large-scale mutation that updates the multi-position of individual simultaneously.

A mutation sample is given in Table 3. X_3 is selected randomly, the flip-flop method is employed, and the new value of X_3 is to be 0.

Table 3 - The example of the mutation operator

	X_1	X_2	X_3	X_4	X_5
Individual	*1*	*0*	***1***	*0*	*1*
New individual	*1*	*0*	***0***	*0*	*1*

Selection strategies

Selection strategies are used to increase the survival probability of individuals and offspring with higher fitness in the next generation and to select parents (Kilic et al., 2021). Roulette Wheel Selection and Tournament Selection are popular selection strategies.

Roulette Wheel Selection: A circular wheel consists of n pies, where n is the solution number in the population. Each solution gets a portion of the pie based on its fitness value (Sharifi & Aghdam, 2019). A point on the wheel circumference is selected and the circular wheel is spun.

Solutions	Fitness Values
A	10
B	15
C	9
D	25
E	30

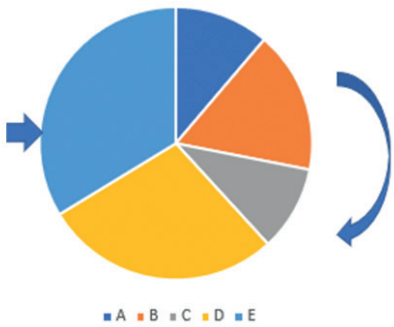


Figure 2 – A sample population

Tournament Selection: In this approach a "tournament, k" selection, k-individuals are selected randomly from the population, and one with the best fitness value among them using a tournament (Blickle, 2000).

9.3 Knapsack Problem

In the knapsack problem, a pack from a set of items with weights and values wants to be put into the knapsack with the maximum total value (Salkin & De Kluyver, 1975, Gaivoronski et al., 2011). Table 4 shows test dataset for the knapsack problem.

Table 4 – Test dataset for the knapsack problem

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7
Weight	30	20	10	45	15	33	25
Value	10	5	30	16	50	13	13
A sample Solution	1	0	1	1	0	1	1

We use the following notation, parameters, and decision variables.

Notation:

j: item index, j ∈ {1...J}, *J is number of items*

Parameters:

v_j: value of item *j*

w_j: weight of item *j*

W: maximum capacity of the knapsack

Decision variables:

$$x_j = \begin{cases} 1, & \textit{if item } j \textit{ is selected into the knapsack} \\ 0, & \textit{otherwise} \end{cases}$$

$$\text{Mazimize F} = \sum_{j=1}^{J} v_j x_j$$

$$\text{subject to} = \sum_{j=1}^{J} w_j x_j < W$$

Fitness Function Code:

```
function Fit = MyFitness(x)
   global wSet vSet maxCapacity;
   sumV = sum(x(1,:).* vSet);
   sumW = sum(x(1,:).* wSet);
   if sumW <= maxCapacity
     Fit= sumV;
   else
     Fit = 0;
   end
```

Genetic Algorithm Code:

```
clc;
```

```
clear;
close all;

global nItem wSet vSet maxCapacity;
wSet = [30, 20, 10, 35, 15, 33, 25, 25, 25, 15, 25,54]; % weights of each item
vSet = [10, 5, 30, 16, 50, 13, 13, 23, 14, 52, 10,50]; % value of each item
maxCapacity = 120;
nItem = size(wSet,2);
FitnessFunction = @(x) MyFitness(x);
WeighFunction = @(x) MyFitnessW(x);
popSize = 20;
maxIter = 50;

muProbility = 0.2;
individual.Solution = [];
individual.FitnessValue = [];
individual.Weight = [];
population = repmat(individual, popSize, 1);
round(rand(1,nItem));
for i = 1:popSize
   population(i).Solution = round(rand(1,nItem));
   population(i).FitnessValue = FitnessFunction(population(i).Solution);
   population(i).Weight = WeighFunction(population(i).Solution);
end

% Sort Population
FitnessValues = [population.FitnessValue];
[FitnessValues, SortOrder] = sort(FitnessValues,'descend');
population = population(SortOrder);

BestSol = population(1);
BestFitness = zeros(maxIter, 1);
TournamentSize=3;

for t = 1:maxIter
   % Crossover operator
   populationCrossover = repmat(individual, popSize/2, 2);
   for j = 1:popSize/2
      i1 = TournamentSelection(population, TournamentSize);
```

```
      i2 = TournamentSelection(population, TournamentSize);
      p1 = population(i1);
      p2 = population(i2);
      % Perform Crossover
      [populationCrossover(j, 1).Solution, populationCrossover(j, 2).Solution] =
      Crossover(p1.Solution, p2.Solution);

   % Evaluate Offsprings
   populationCrossover(j, 1).FitnessValue = FitnessFunction(populationCrossover
   (j, 1).Solution);
   populationCrossover(j, 2).FitnessValue = FitnessFunction(populationCrossover
   (j, 2).Solution);
   populationCrossover(j, 1).Weight = WeighFunction(populationCrossover
   (j, 1).Solution);
   populationCrossover(j,  2).Weight = WeighFunction(populationCrossover
   (j, 2).Solution);
end

   populationCrossover = populationCrossover(:);

   % Mutation operator
   mutPop =0;
   populationMutation = repmat(individual, 0,1);
   for j = 1:popSize
      p = population(i);
   if (rand < muProbility)
      mutPop=mutPop+1;
      k= randi(nItem);
      p.Solution(k) = 1- p.Solution(k);
      p.FitnessValue = FitnessFunction(p.Solution);
      p.Weight = WeighFunction(p.Solution);
      populationMutation(mutPop) = p;
   end
end

populationMutation = populationMutation(:);
   population = [population
   populationCrossover
   populationMutation];
```

```
FitnessValues = [population.FitnessValue];
[FitnessValues, SortOrder] = sort(FitnessValues,'descend');
population = population(SortOrder);
population = population(1:popSize);
FitnessValues = FitnessValues(1:popSize);

BestSol = population(1);

BestFitness(t) = BestSol.FitnessValue;
disp(['Generation : ' num2str(t) ': Best Fitness value = ' num2str(BestFitness(t))]);
end

plot(1:maxIter,BestFitness);
```

CHAPTER 10

SINGLE LAYER NEURAL NETWORK (PERCEPTRON)

Neural networks (NNs) store information in the form of weights learned either from the supervised (pattern recognition) or unsupervised (function approximation) learning perspectives. NNs are essentially non-parametric modeling approaches used for approximate representation of the real systems. Therefore, its analytic (insightful and rigorous mathematical) analysis is challenging. In order to train the NNs, the weights should be updated based on the information provided through the inputs. The systematic approach used for weights update is the called the learning rule which utilizes the provided input information. It essentially maps the input information to the output information. Since the training is the only way for the NNs to store and remember the information systematically, the learning rule is a vital component of the learning process, discussed next.

10.1 Delta Rule

The delta rule is a representative learning rule of the single layer NNs. The training process of a single layer NN can be shown as in the figure below.

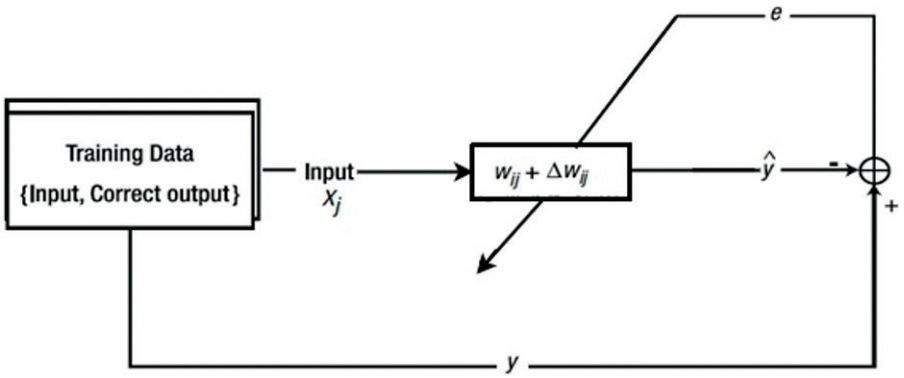


Figure 1: Block diagram of a single-layer NN training process

Important to note that the single layer NN can be single input single output (SISO), single input multiple output (SIMO), multiple input single output (MISO) or multiple input and multiple output (MIMO). The number of the inputs and outputs vary depending on the character of the learning problem. Also note that coupled dynamics can only be learned by the NNs having more than one inputs or outputs. However, if the learning problem is not coupled, but the learning problem of NN is

constructed as coupled, then the efficiency of the NNs will reduce. Therefore, character of the input data should initially be analyzed, and based on the gained insights about the input data, the NNs should be constructed.

Pseudo Code for delta learning rule for *m* Inputs and *n* Outputs in a neural network:

1. Input: Training input $x \in \square^{m \times l}$, where l is the length of each m number of the input data, labelled output $y \in \square^{l \times n}$, where n is the number of the output, randomly initialized unknown parameter matrix/vector $w \in \square^{m \times n}$, estimated output $\hat{y}_o \in \square^{n \times l}$, saturated and estimated $\hat{y} \in \square^{n \times l}$, the training error $e \in \square^{l \times n}$, the parameter learning rate $0 < \eta \le 2 / x(:,1)^T x(:,1)$, the number of the multiple simulations simMultiple, store matrix w_s, store error e_s, error stopping threshold e_t, store estimated output $\hat{y}_s$.
2. Output: The terminal value of the trained parameters w, store learning error e_s, store output $\hat{y}_s$
3. for *i* to simMultiple
4. for *j* to *l*
5. 1. Calculate the estimated current output $\hat{y}_o$
6. $\hat{y}_o(:, j) = w^T x(:, j)$
7. 2. Apply a threshold σ to the output (if necessary)
8. $\hat{y}(:, j) = \sigma(\hat{y}_o(:, j))$
9. 3. Determine the error
10. $e(:, j) = y - \hat{y}(:, j)$
11. 4. Update and store the unknown parameter
12. $w \to w + \eta e(:, j) x(:, j)^T$
13. $w_s = [w_s; reshape(w_s, 1, [])]$
14. end *j*
15. Store the error and the saturated output
16. $e_s = [e_s; reshape(e, 1, [])]$
17. $\hat{y}_s = [\hat{y}_s; reshape(\hat{y}, 1, [])]$
18. If $e(:, j) < e_t$ then
19. break
20. end if
21. end *i*

The delta rule updates the unknown parameters iteratively rather than solving it all at once. It is a type of numerical iterative method using gradient descent. The gradient descent starts from the initial value and proceeds towards the solution. Its name originates from its behavior whereby it searches for the solution as if a ball rolls down the hill along the steepest path. In this analogy, the position of the ball is the occasional output from the model, and the bottom is the solution. It is noteworthy that the iterative gradient descent method cannot drop the ball to the bottom with just one throw. The whole

process repeats, as retraining the model with the same data may improve the model.

Example: Delta Rule

Consider a NN that consists of three input nodes and one output node, as shown in the following figure.

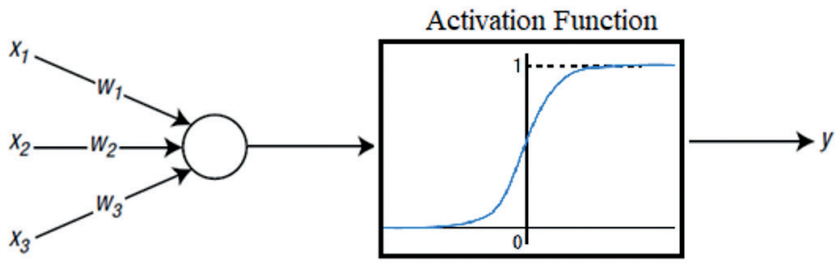


Figure 2: The NN that consists of three input nodes and one output node

As can be seen from Figure 2, the sigmoid function is used for the activation function of the output node. We have four training data points as shown in the following table.

Table 1 - OR Gate training data points with labels

{0,0,1,**0**}
{0,1,1,**1**}
{1,0,1,**1**}
{1,1,1,**1**}

As they are used for supervised learning, each data point consists of an input-correct output pair. The last bold number of each dataset is the correct output. This is an OR Gate problem having the last value of the input as the bias of 1.

As it is single layered and contains simple training data, the code is not complicated. Once you follow the code, you will clearly understand the learning behavior of the NN.

The corresponding code proceeds as follows:

Initially, training parameters are defined as in the following "trainPar"function:

```
function trainPar = trainParameters()
% Training input data where the last value of 1 represents the bias
trainPar.x = [0 0 1; 0 1 1; 1 0 1; 1 1 1]';
% Labelled output data

trainPar.y = [0 1 1 1]';
% Randomly intialized unknown parameters
```

```
trainPar.w = rand(size(trainPar.x,1),size(trainPar.y,2));
% Intitialize the estimated output
trainPar.yo_hat = zeros(size(trainPar.x,2),size(trainPar.y,2));

% Intitialize the estimated output
trainPar.y_hat = zeros(size(trainPar.x,2),size(trainPar.y,2));

% Intitialize the error
trainPar.e = zeros(size(trainPar.x,2),size(trainPar.y,2));

% Intitialize the learning rate
trainPar.mu = zeros(size(trainPar.y));

% Learning rate upper scaling
trainPar.mur = 2;

% Stopping error threshold
trainPar.et = 0.001;

% The number of the multiple trainings
trainPar.simMultiple = 1000;

% The output saturation function upper limit (sigmoid)
trainPar.satUppper = 1;

end
```

After defining the related training parameters, the following function is used for learning process.

```
% This m-file trains a single layer NN for the OR problem
% Upload the training parameters

trainPar = trainParameters();
% Upload the allocated error

e = trainPar.e;
% Upload the allocated estimated output

yo_hat = trainPar.yo_hat;
 Upload the allocated ouput with threshold

y_hat = trainPar.y_hat;
 Upload the allocated unknown parameter

w = trainPar.w;
% Upload the allocated learning rate

mu = trainPar.mu;
% Introduce the store matrix for the unknown parameter

ws = [];
% Introduce the stora matrix for the error
```

```
es = [];
% Introduce the store matrix for the estimated output
ys_hat = [];
for i=1:trainPar.simMultiple
   for j=1:size(trainPar.x,2)
      % Calculate the estimated current output
      yo_hat(j,:) = w'*trainPar.x(:,j);
      % Apply a threshold for the estimated output
      y_hat(j,:) = satOutput(yo_hat(j,:),trainPar);
      % Determine the instant error
      e(j,:) = trainPar.y(j,:) - y_hat(j,:);
      % Update the learning rate
      mu(i,j) =trainPar.mur/(trainPar.x(:,j)'*trainPar.x(:,j));
      % Update the unknown parameter vector/matrix
      w = w + mu(i,j)*e(j,:)*trainPar.x(:,j);
      % Store the unknown parameter vector/matrix
      ws = [ws;reshape(w,1,[])];
end
% Store the error history
es = [es;reshape(e,1,[])];
% Store the estimated output
ys_hat = [ys_hat;reshape(y_hat,1,[])];
end
```

where ***satOutput*** *function is created for sigmoid activation function as given below:*

```
function y_sat = satOutput(y_unsat, trainPar)
y_sat = trainPar.satUppper / (1 + exp(-y_unsat));
end
```

Then the estimated outputs **ys_hat** *for each set of inputs is plotted using the following code block:*

```
figure(1),
plot(1:length(ys_hat),ys_hat(:,1),'r','LineWidth',2),
hold on,
plot(1:length(ys_hat),ys_hat(:,2),'b','LineWidth',2),
```

```
plot(1:length(ys_hat),ys_hat(:,3),'g','LineWidth',2),
plot(1:length(ys_hat),ys_hat(:,4),'y','LineWidth',2),
hold off
title('Estimated Outputs for the OR Gate')
```

Performing this code produces the following figure:

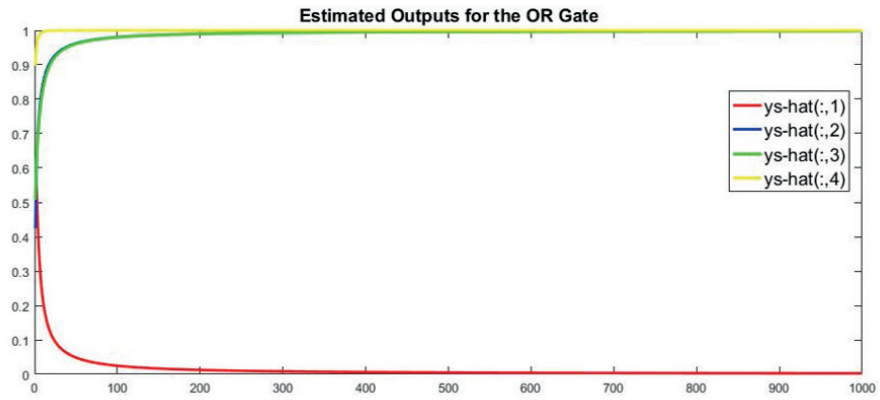


Figure 3: Estimated outputs results for the OR Gate

Executing this code produces the following values. These output values are very close to the correct outputs in target y value. Therefore, we can conclude that the NN has been properly trained to learn OR Gate.

$$\begin{bmatrix} 0.0025 \\ 0.9980 \\ 0.9980 \\ 1.0000 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

To plot the training error,

```
figure(),
plot(1:length(es),es(:,1),'r','LineWidth',2),
hold on,
plot(1:length(es),es(:,2),'b','LineWidth',2),
plot(1:length(es),es(:,3),'g','LineWidth',2),
plot(1:length(es),es(:,4),'y','LineWidth',2),
hold off
title('Training Error for the OR Gate')
```

code block is used and determined result is represented as:

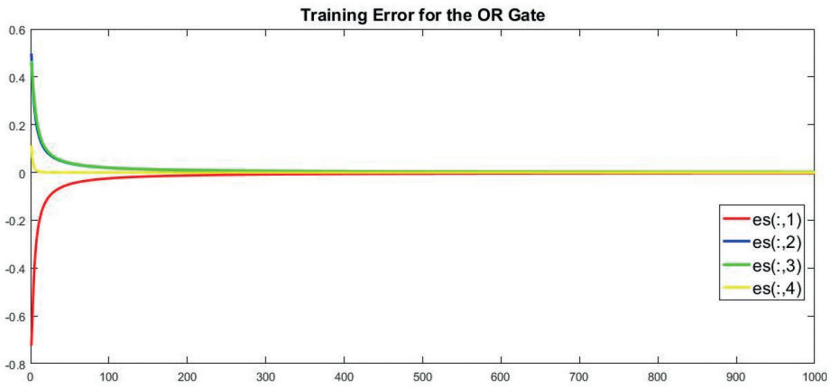


Figure 4: Training Error results for the OR Gate

As can be seen from Figure 4, the error converges to zero for corresponding OR Gate data points, respectively.

Finally, trained unknown parameters are plotted and shown using the following code block:

```
figure(),
plot(1:length(ws),ws(:,1),'r','LineWidth',2),
hold on,

plot(1:length(ws),ws(:,2),'b','LineWidth',2),
plot(1:length(ws),ws(:,3),'g','LineWidth',2),
plot(1:length(ws),ws(:,4),'y','LineWidth',2),
hold off
title('Trained Unknown Parameters for the OR Gate')
```

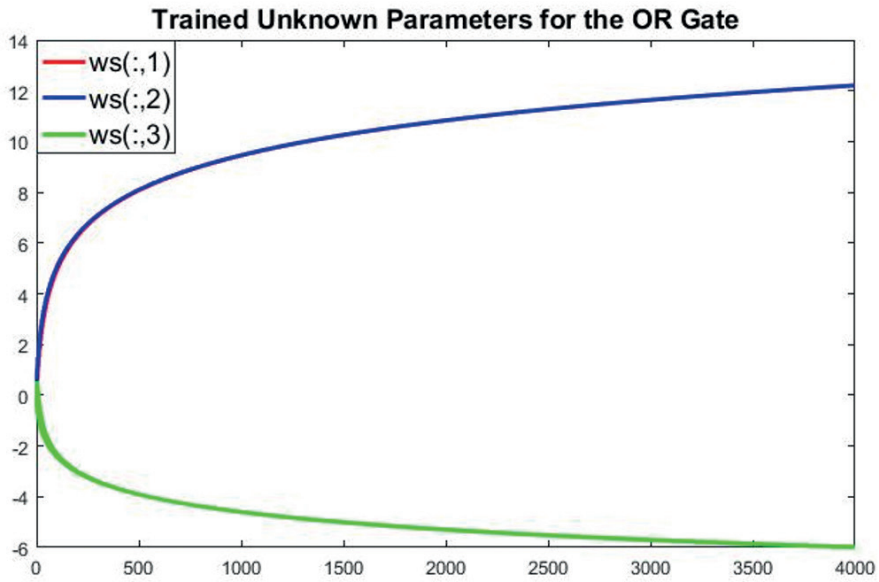


Figure 5: Trained Unknown parameters for the OR Gate

As clearly seen in Figure 5, only three trained unknown parameters are plotted. This occurred due to the multiplication of matrices in the following code block with each other.

```
trainPar.w = rand(size(trainPar.x,1),size(trainPar.y,2));
```

where size(trainPar.x) is 3x4 and size(trainPar.y) is 4x1. Thus, a 3x1 vector will be determined. In fact, it is quite simple to draw all the trained unknown parameters here. After all these explanations, please make the required update in the code.

10.2 Limitations of Single Layer NNs

This section presents the critical reason that the single layer NN had to evolve into a multi-layer NN. We will try to show this through a particular case. Consider the same NN that was discussed in the previous section. It consists of three input nodes and one output node, and the activation function of the output node is a sigmoid function. Assume that we have four training data points as shown below.

Table 2: XOR gate training data points with labels

{0,0,1,**0**}
{0,1,1,**1**}
{1,0,1,**1**}
{1,1,1,**0**}

As given in Table 2, this is an XOR Gate problem having the last value of the input as the bias of 1. It is different from the 'Delta Rule' section in that the second and fourth correct outputs are switched while the inputs remain the same. Well, the difference is barely noticeable.

As we are considering the same NN, we can train it using the "trainPar" function from the "Example: Delta Rule" section, except that it has different values for y as mentioned before. Before executing the code, the labeled output data code block in the "trainPar" function is updated as follows.

```
% Labelled output data
trainPar.y = [0 1 1 0]';
```

Executing this code, the following values will appear, which consist of the output from the trained NN corresponding to the training data. We can compare them with the correct outputs given by "y" as:

$$\begin{bmatrix} 0.5297 \\ 0.5000 \\ 0.4703 \\ 0.4409 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

As can be seen from the determined equation, we got two totally different sets. Training the NN for a longer period does not make a difference.

What actually happened?

Illustrating the training data can help elucidate this problem. Let's interpret the three values of the input data as the X, Y, and Z coordinates respectively. As the third value (Z coordinate) is fixed as 1, the training data can be visualized on a plane as shown in the following figure.

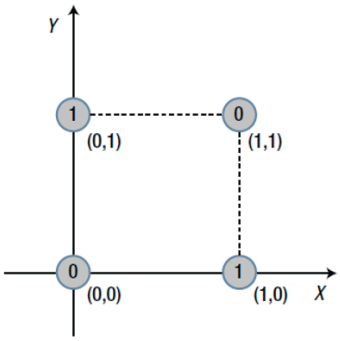


Figure 6: Interpreting the three values of the input data as the X, Y, and Z coordinates

The values 0 and 1 in the circles are the correct outputs assigned to each point. One thing to notice from this figure is that we cannot divide the regions of 0 and 1 with a straight line. However, we may divide it with a complicated curve as shown from the following figure. This type of problem is said to be linearly inseparable.

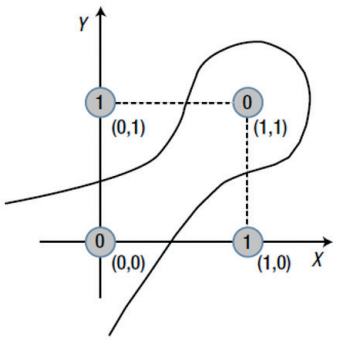


Figure 7: Separating 0 and 1 with a complicated curve (linearly inseparable)

In the same process, the training data from the "Example: Delta Rule" section on the X-Y plane appears as:

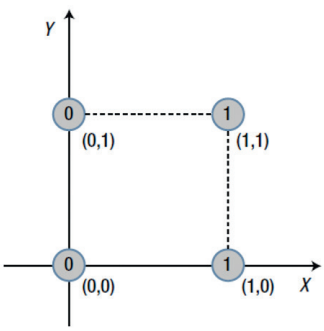


Figure 8: The delta rule training data

In this case, a straight border line that divides the regions of 0 and 1 can be found easily. This is a linearly separable problem as shown in the following figure:

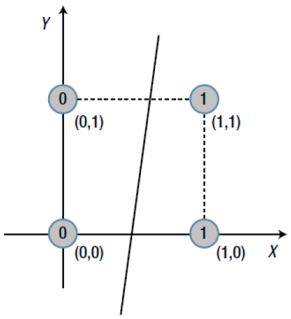


Figure 9: Linearly separable problem

To put it simply, the single layer NN can only solve linearly separable problems. This is because the single layer NN is a model that linearly divides the input data space. In order to overcome this limitation of the single layer NN, we need more layers in the network. This need has led to the appearance of the multi-layer NN which can achieve that the single layer NN cannot. Just keep in mind that the single layer NN is applicable for specific problem types. The multi-layer NN has no such limitations. Please see the references below for more details.

CHAPTER 11

NEURAL NETWORK IMPLEMENTATION

This part of the handbook was written by Jarmila Škrinárová from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

Matlab is a high-level language and interactive environment for numerical computation, visualization, and programming for:

- data analysis,
- algorithm development,
- model and application creation.

The goal of the section is to learn how to work with Matlab and how to create simple neural networks. We introduce a methodology and three neural network examples in a graphical Matlab environment. The examples are oriented on the creation fitting function and classification tasks.

11.1 Brief introduction to Matlab – MAtrix LABoratory

First, we introduce the Matlab environment. There is a tool strip on the upper side of the window. Below the toolbar, the area is divided into four windows, which are intended for navigation (moving through the directory structure), editing executable scripts, displaying the workspace, and a command window (see Figure 1).

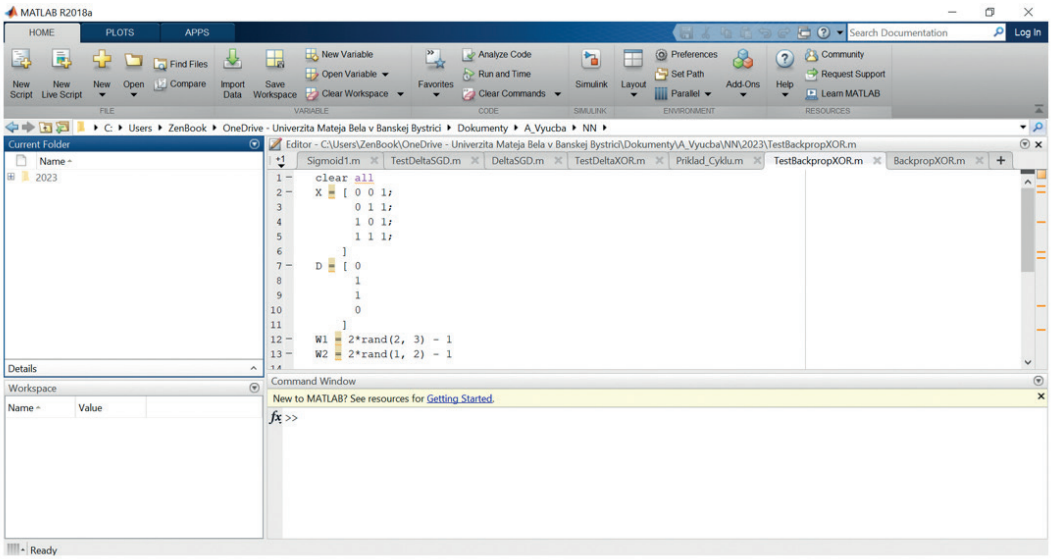


Figure 1: Editor, Command window, workspace, navigator

First, we will learn to work in the command window, where we write commands after the ">>" sign (see Figure 2).

Figure 2: Command window

Examples of simple computations, work with variables, vectors, and matrices

Computations with **variables** – examples 1 to 4 can be gradually practiced directly in the command window:

Example 1	Example 2	Example 3	Example 4
>> 12+34 ans = 46	**>> a =5, b = a^2** a = 5 b = 25	**>> (101+79) /(47 -17)** ans = 6	**>> 15*12** ans = 180

A **vector** is a one-dimensional array of elements. We usually write the individual elements of vectors in square brackets and separate them with a comma or a space. Note that we write the note after the % sign. In the following parts of this chapter, we will work with neural networks, and we need input and target data for neural network learning. If the network has one input, the input data is a one-dimensional array of elements (vector). If the network has one output, the target data is also a one-dimensional array of elements [3].

Example 5:	Example 6:	Example 7:	Example 8:
>> **u1=[1 2 3 4]** %row vector u1 = 1 2 3 4	>> **u2=[1 2 1 2]** %row vector u2 = 1 2 1 2	>> **u1.*u2** %scalar product of two vectors ans = 1 4 3 8	>> **v=[-1; -7; -3]** %column vector v = -1 -7 -3

Example 9:	Example 10:	Example 11:	Example 12:
>> **w=[1 7 -2]'** %transposed vector w = 1 7 -2	>> **6:2:12** % To generate a regular vector, we define the first and last elements of the vector and the step. ans = 6 8 10 12	>> **m=15:-3:0** m =15 12 9 6 3 0	>> **x=12** x = 12 >> **z=[x, 2*x, 3*x]** z = 12 24 36

Example 13:	Example 14:
>> **W=2*rand(1,3)-1** W = 0.9298 -0.6848 0.9412	>> **x2=linspace(-1, 4, 8)** % -1 to 4 is interval and 8 is number of elements x2 =-1.0000 -0.2857 0.4286 1.1429 1.8571 2.5714 3.2857 4.0000

To use more than one input or target in neural networks, we need to prepare data in the form of two-dimensional fields. In Matlab, two-dimensional arrays are represented by **matrices**. Therefore, we will practice working with matrices:

Example 15:	Example 16:
>> **A=[1 -1 2 -3; 3 0 4 5; 3.2, 5 -6 12]** %matrix A = 1.0000 -1.0000 2.0000 -3.0000 3.0000 0 4.0000 5.0000 3.2000 5.0000 -6.0000 12.0000	>> **O=[]** %empty matrix O = []

Example 17:	Example 18:
>> **B=[A; u1]** %Matrix expansion by 1 row (vector u1). B = 1.0000 -1.0000 2.0000 -3.0000 3.0000 0 4.0000 5.0000 3.2000 5.0000 -6.0000 12.0000 1.0000 2.0000 3.0000 4.0000	>> **C=[A, v]** %Extending the matrix by 1 column (vector v). C = 1.0000 -1.0000 2.0000 -3.0000 -1.0000 3.0000 0 4.0000 5.0000 -7.0000 3.2000 5.0000 -6.0000 12.0000 -3.0000

Example 19:	Example 20:
>> **Z=zeros(2,5)** %Creating a null matrix of size 2 rows by 5 columns. Z = 0 0 0 0 0 0 0 0 0 0	>> **O1=ones(3,4)** %Creating a unit matrix with dimension 3 rows by 4 columns. O1 = 1 1 1 1 1 1 1 1 1 1 1 1

Example 21:	Example 22:
>> **A=[1 -1 2 -3; 3 0 4 5; 3.2, 5 -6 12]** A = 1.0000 -1.0000 2.0000 -3.0000 3.0000 0 4.0000 5.0000 3.2000 5.0000 -6.0000 12.0000 >> **A(2, :)** % Listing of the 2nd row of matrix A ans =3 0 4 5 >> **A(:, 3)** % Listing of the 3rd column A matrix ans = 2 4 -6	>> **I=eye(5,8)** %Creating a diagonal matrix of size 5 rows by 8 columns. I = 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0
Example 23:	Example 24:
>> **R1=rand(3,5)** %Create a random matrix of size 3 rows by 5 columns, with values in the range 0 to 1 R1 = 0.1419 0.7922 0.0357 0.6787 0.3922 0.4218 0.9595 0.8491 0.7577 0.6555 0.9157 0.6557 0.9340 0.7431 0.1712	>> **R2=randn(4)** %matrix with random elements - standard distribution R2 = 0.8884 -2.9443 1.3703 0.3192 -1.1471 1.4384 -1.7115 0.3129 -1.0689 0.3252 -0.1022 -0.8649 -0.8095 -0.7549 -0.2414 -0.0301

Example 25:	Example 26:
>> **A=[1 5 0; -1 2 3; 1 2 1]** A = 1 5 0 -1 2 3 1 2 1 >> **c=2** c = 2 %matrix multiplication by vector >> **D=A*c** D = 2 10 0 -2 4 6 2 4 2	>> **B=[1 2 3; 1 2 3; 1 2 3]** B = 1 2 3 1 2 3 1 2 3 >> **E=B*D** %matrix multiplication E = 4 30 18 4 30 18 4 30 18
Example 27:	Example 28:
>> **A =[2 3; 0 10]** B =[1 0; -3 5] %matrix multiplication A = 2 3 0 10 B = 1 0 -3 5 >> **C=A*B** C = -7 15 -30 50	%matrix scalar multiplication, matrices A and B are from previous example >> **C=A.*B** C = 2 0 0 50

We often use data that contains many elements. Therefore, it is practical if we write this data to a **file** so that we can use it later. All the data we have worked with in Matlab are stored in the workspace and can be seen in the lower left window. We can display information about the workspace's contents using examples 29 and 30.

Example 29:	Example 30:
>> **who** % workspace listing only with names of variables, vectors and matrices Your variables are: A B C O ans u v w x z	>> **whos** % workspace Name Size Bytes Class Attributes A 3x4 96 double B 4x4 128 double C 3x5 120 double O 0x0 0 double ans 1x1 8 double u1 1x4 32 double u2 1x4 32 double v 3x1 24 double w 3x1 24 double x 1x1 8 double z 1x3 24 double

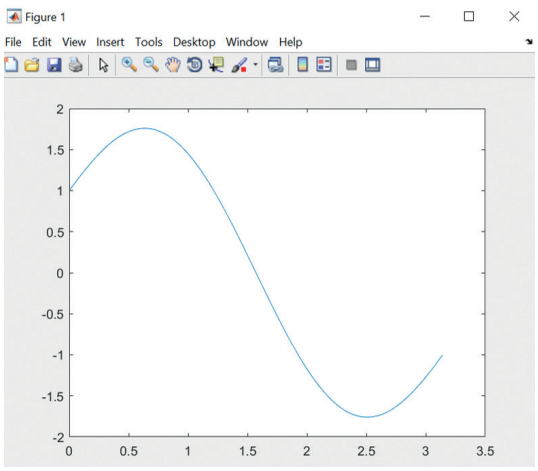
We can **save** all variables, vectors, and matrices **from the workspace** to the file (see example 31) or only selected variables, vectors, and matrices (see example 32).

Example 31:	Example 32:
>> **save data** % save all data to file data.dat	>> **save data1 u1 u2 v** % save variables u1, u2 and v into data1.mat

Data stored in files can be loaded anytime into the workspace of the Matlab tool to work with it. See examples 33 and 34.

Example 33:	Example 34:
>> **load data1** % read all variables saved in data1.mat	>> **load data.dat -MAT** % read all variables saved in data.dat

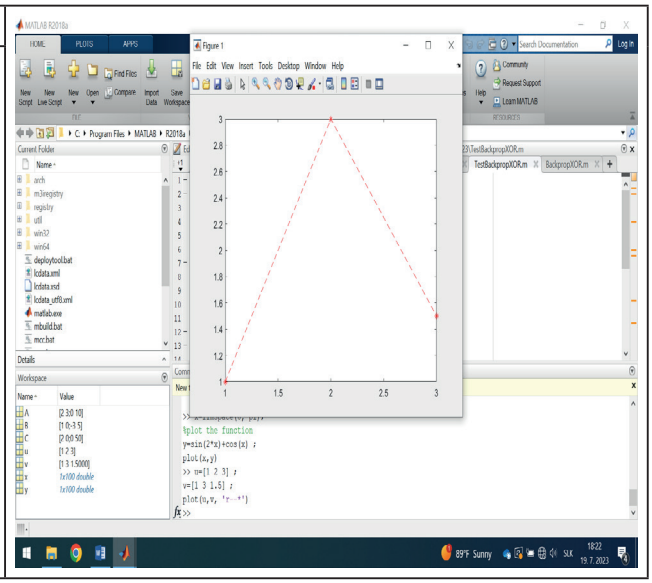
When **plotting** the course of a particular function, the number of elements on the x-axis must be the same as on the y-axis. We can see the drawn function in the picture in example 35. The function is drawn in a new window that can be edited - insert axis names, titles, etc.

Example 35:	
>> x=linspace(0, pi); %plot the function **>> y=sin(2*x)+cos(x) ;** **>> plot(x,y)**	

Both the **color** and the **line type** can be changed in the plot command. See example 36.

Example 36:

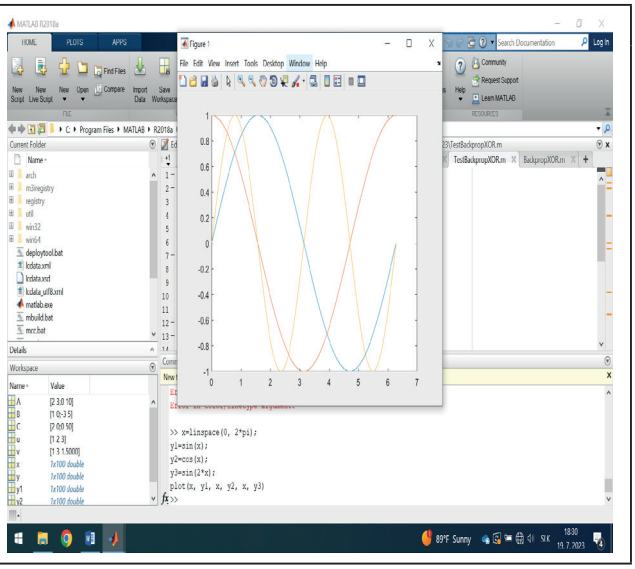
```
>> u=[1 2 3] ;
>> v=[1 3 1.5] ;
>> plot(u,v, 'r--*')
```



It is also possible to **plot several functions** in one image. See example 37.

Example 37:

```
>> x=linspace(0, 2*pi);
>> y1=sin(x);
>> y2=cos(x);
>> y3=sin(2*x);
>> plot(x, y1, x, y2, x, y3)
```



For illustration, we show a simple program example (example 38), where individual rows of the matrix X are sequentially written out in a cycle. The output from the program is in the right column.

Example 38:	
>> clear all X = [0 0 1; 0 1 1; 1 0 1; 1 1 1;]; N = 4; % print always one row of matrix X for k = 1:N x = X(k, :) end	x = 0 0 1 x = 0 1 1 x = 1 0 1 x = 1 1 1

In the previous sections, we showed how to write commands in the command line. This is not practical if we need to write many commands. We write such successive commands in a text editor and save them in a file with the extension "m." This is how we create a **script** to open and run in the Matlab environment. The commands we write in the script should first be tested in the Matlab command line to avoid errors.

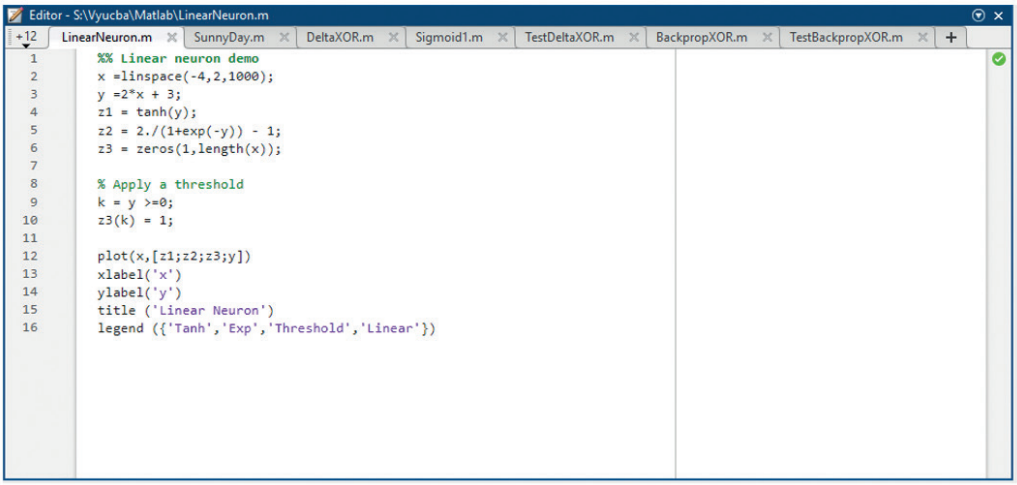


Figure 3: Example of M-file in Matlab editor window

11.2 Implementation of neural networks in Matlab

In the real world, it often happens that we can make various measurements, but we cannot describe the behavior of a particular system with a simple mathematical model. This means that we have the measured values of the inputs to the system and the corresponding outputs, but we cannot calculate the outputs based on the inputs. For this purpose, we use neural networks to learn the relationship between inputs (from a certain interval of values) and outputs or classify inputs into certain groups. A well-trained neural network can produce correct outputs for different input values (from the same interval).

This subsection aims to specify the **methodology** for creating neural networks in the Matlab environment. Subsequently, we will solve **simple examples** to help us understand the neural network creation process.

Methodology of neural network creation in Matlab graphical environment

We will describe the methodology in individual steps:

Step 1: Data preparation. We usually need two sets of data (input set and their corresponding output set). If we have one data sample where one input corresponds to one expected output (target), then both datasets have the same size, i.e., 1 row x number of columns (samples).

Step 2: Select a suitable application from the APPS tab in the Matlab environment. For example, from the Machine Learning category, select the Neutral Net Fitting application (see Figure 4).

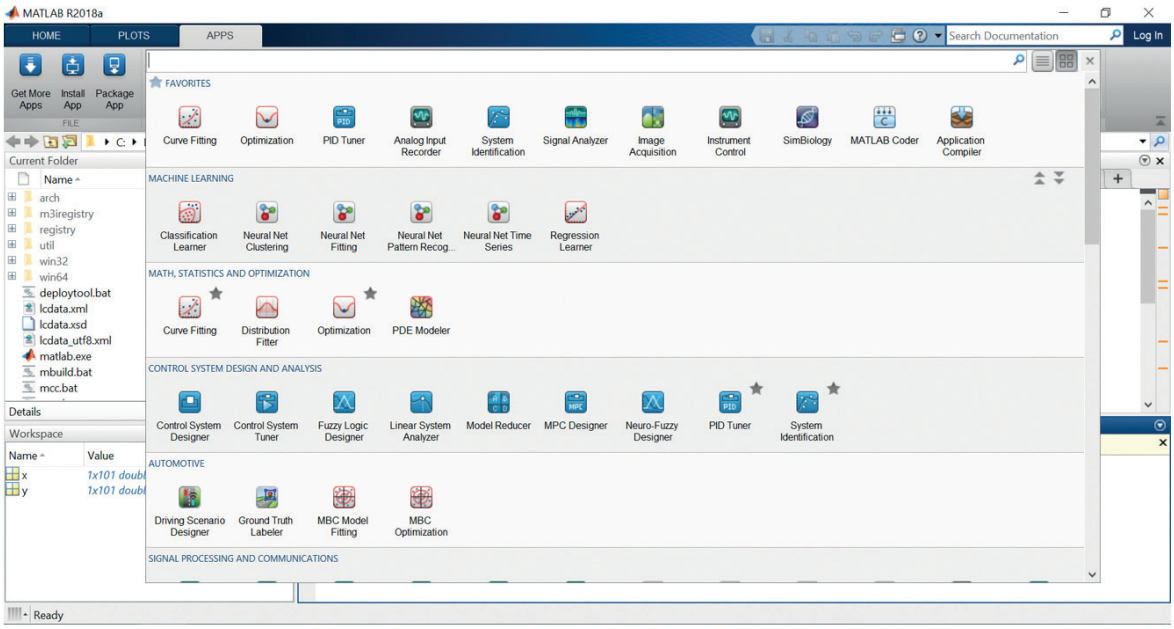


Figure 4: Applications that are part of the Matlab environment

Step 3: We load the input data from the dataset and the target data from the dataset (those we prepared in Step 1).

Step 4: We enter the ratio in which the data should be divided into three sets for training, validation, and testing, for example, 70%, 15%, and 15%.

Step 5: We will design the network architecture. The number of network inputs and outputs is automatically set according to input and target data. It is necessary to set the number of neurons in the hidden layers. For example, suppose we are designing a multilayer perceptron network that has 2 hidden layers. In that case, we need to specify the number of neurons for the first and second hidden layers.

Step 6: Selection of the learning algorithm. We choose one of the prepared learning algorithms, for example, Levenberg - Marquardt, Bayesian Regularization, or scaled conjugate gradient.

Step 7: We start the network learning process. It should be noted that certain values are preset in the application. For example, the number of learning epochs can be set to 1000, and the learning accuracy is expressed using MSE and R. Mean square error (MSE) is the average squared difference between network outputs after and targets before the training process. Our goal is to get the smallest values of errors. A zero value means no error. Regression (R) expresses the measured correlation between

outputs and targets. An R-value of 1 indicates a close correlation, and 0 means no correlation, or in other words, there is a random relationship.

Step 8: The network learning process ends if the achieved learning accuracy is sufficient for us. Otherwise, the network architecture must be changed (the number of hidden layers and the number of neurons in them), or change the learning algorithm, or change the number of learning epochs of the network if possible. This means that we repeat the procedure from step 5. It should be considered that a high number of learning epochs can lead to so-called network relearning.

Example of simple neural network function creation

In this example, we will show how a neural network learns the value of a function. We follow the methodology presented in section 2.1 of this chapter.

Step 1: We will not use the measured data for simplicity, but we will create the input and output data in Matlab. Using commands in Matlab, we will create two datasets (see code below). The first dataset contains inputs named data1.mat contains values of inputs, and the second dataset is named data2. mat contains values of targets, i.e. expected outputs after learning the network. Elements of inputs and targets are arranged in such an order that individual input elements correspond to respective target elements [2].

```
>> x=0:0.1:10
>> y=2*x.^3
>> plot(x,y)
>> save data1 x
>> save data2 y
```

After running the commands from Listing 1, the values of the x and y vectors are written out, the function graph is drawn (see Figure 5), and the datasets are saved in the files.

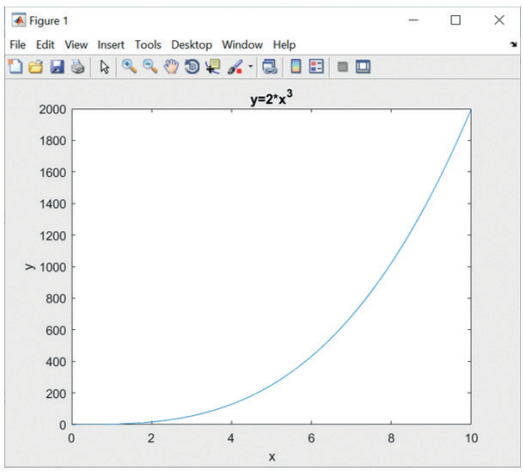


Figure 5: Graph of the function $y = 2x^3$

Step 2: In the Matlab environment, we choose a suitable application from the APPS tab. From the Machine Learning category, we choose the Neutral Net Fitting application. Let's start the Neutral Net Fitting application (see Figure 6). We navigate in the application using the next button.

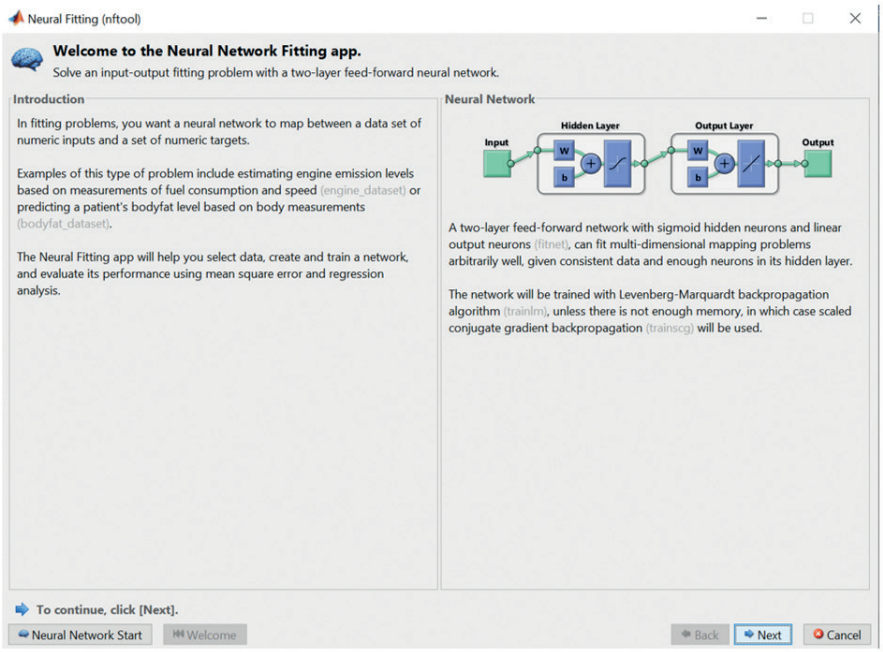


Figure 6: Neural Net Fitting in Matlab

Step 3: We load datasets of input and target data from prepared files (see Figure 7, left). Since we have the same number of inputs and targets, we ensure the files are the same size (see Figure 7, right).

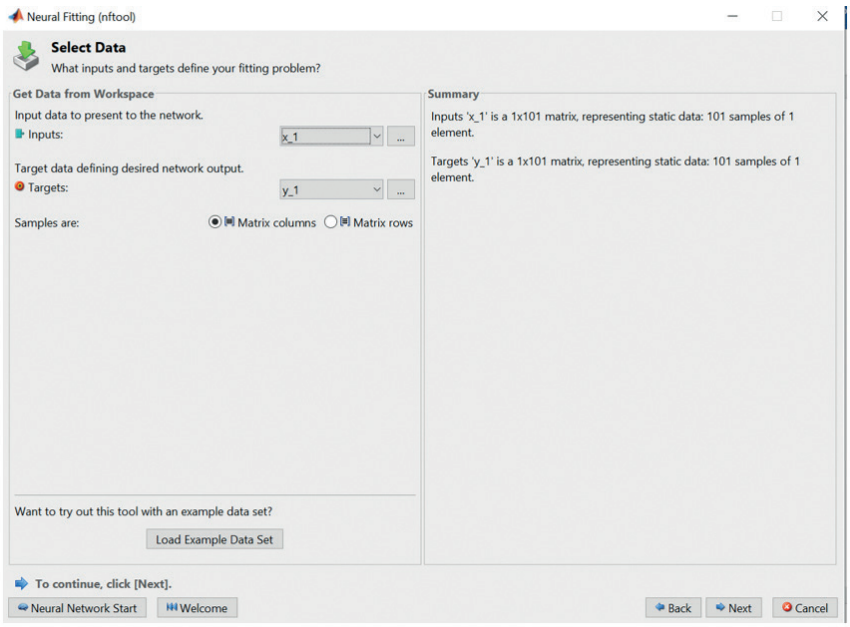


Figure 7: Method of loading input and target values

Step 4: We enter the ratio in which the data should be divided into three sets for training, validation, and testing. In our case, it is 70% of the data for training the network, 15% for validation within the network's learning process, and 15% for testing (see Figure 8).

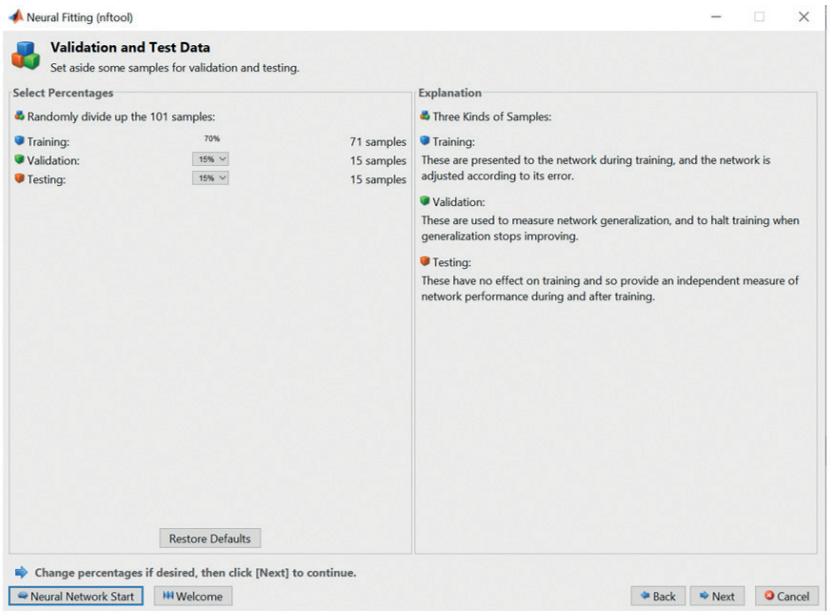


Figure 8: Method of distribution of data samples in the solved task

Step 5: We will design the network architecture. The number of inputs and outputs of the network was automatically set according to the input and output data so that our network has one input and one target (see Figure 9). The output layer has only one neuron because we have one output from the network. The number of neurons in the output layer is also set automatically. In our case, we only have one hidden layer because we use the Neutral Net Fitting application. We set the number of neurons in the hidden layer to 100. If the network does not learn the relationship between inputs and targets accurately enough, we can go back to setting the network architecture and change the number of hidden neurons.

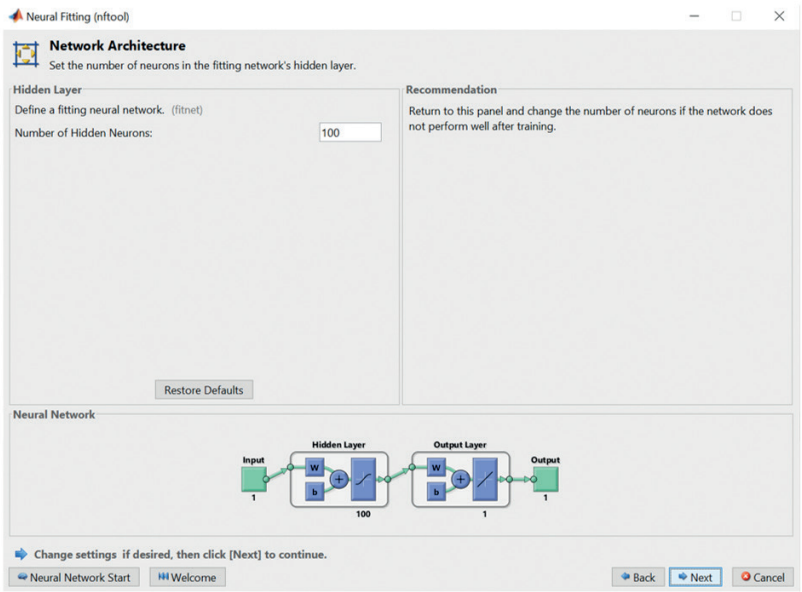


Figure 9: Network architecture design for our task

Step 6: Selection of learning algorithm. We can choose one of three learning algorithms: Levenberg – Marquardt, Bayesian Regularization or Scaled conjugate gradient. We will choose the Bayesian Regularization algorithm (see Figure 10).

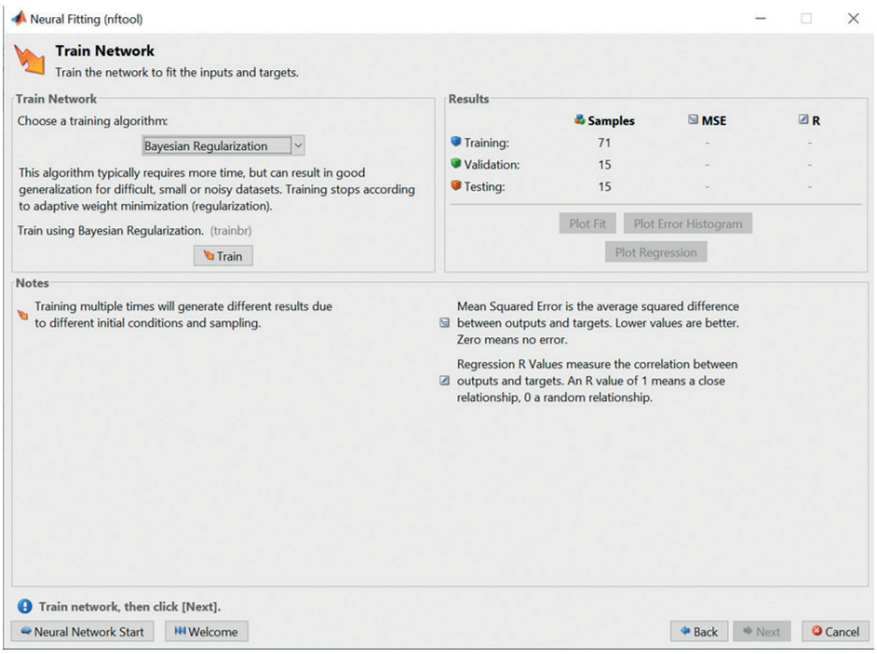


Figure 10: Selection of learning algorithm

Step 7: We start the process of learning the network by pressing the Train button. The number of learning epochs is set to 1000, and we can follow the learning process as shown in Figure 11.

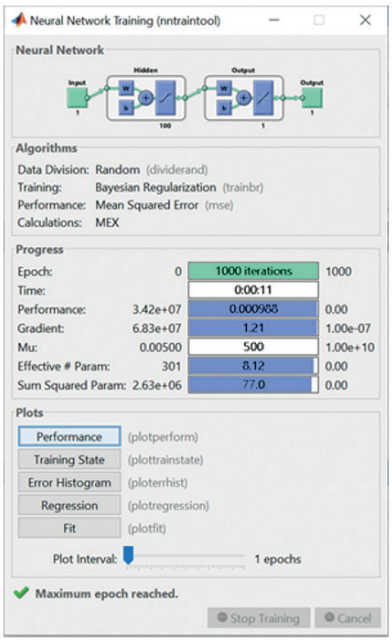


Figure 11: Network learning progress

Learning accuracy is expressed using MSE and R (see Figure 12). Mean square error (MSE) is the average squared difference between network outputs after and targets before the training process. The goal is to get the smallest values of errors. A zero value means no error. We can see that the network learned our feature with an MSE error $8.95*10^{-7}$, which is an insignificant error. Testing the network confirmed that the network learned correctly with a low MSE error of $3.14*10^{-5}$. Regression (R) expresses the measured correlation between outputs and targets. An R-value of 1 means a close correlation, and 0 means no correlation or there is a random relationship. The calculation of correlations after training the network and testing the network reached the value of 1, confirming that the network learned the relationship between the network inputs and outputs very well.

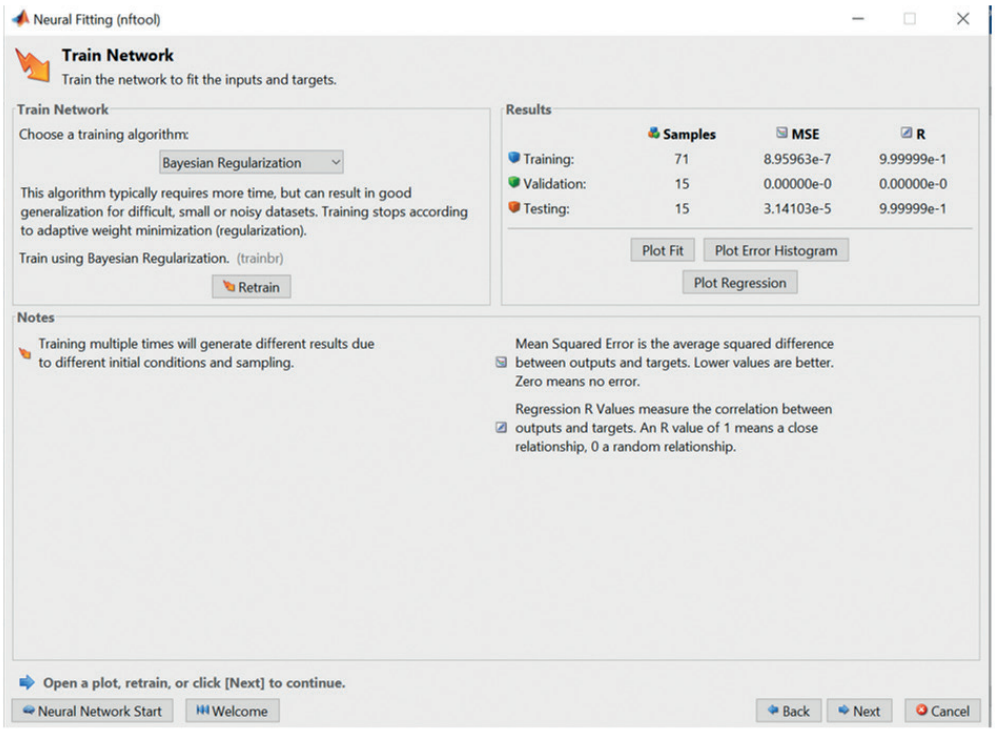


Figure 12: Errors and correlations after the network learning and testing process

Step 8: We can conclude that the achieved accuracy of learning and testing the network is sufficient, and the learning process of the network ends. Therefore, we can see the values and progress of the learned function in more detail if we successively press the buttons Plot Fit, Plot Error Histogram, and Plot Regression (see Figure 12). After pressing Plot Fit, we know the progress of the values of targets and outputs of the network depending on the inputs after learning and testing the network (see Figure 13).

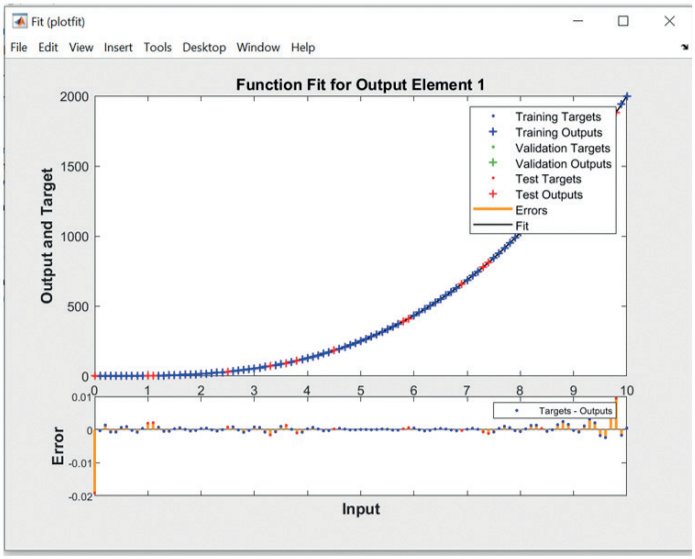


Figure 13: The progress of the targets and outputs of the network depending on the inputs after the process of learning and testing the network

After pressing **Plot Error Histogram**, we can see the error values and their frequency (see Figure 14). In this case, the absolute error is the difference between the target value and the network output about a specific network input. We can see that the error 0.000119 occurred most often.

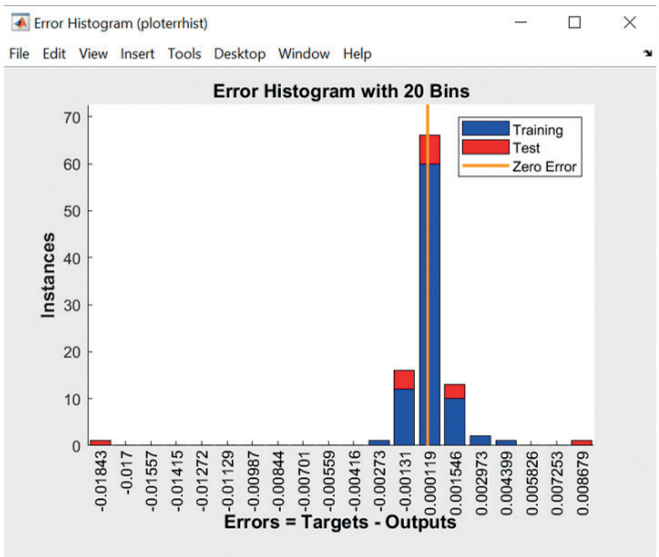


Figure 14: Values of absolute errors and their frequency

After pressing **Plot Regression**, we see the correlation values between the **target** values and the output **values in** the training process, the testing process, and both processes (see Figure 15). The calculation of correlations after training the network and testing the network reached the value of 1, confirming that the network learned the relationship between the network inputs and outputs very well.

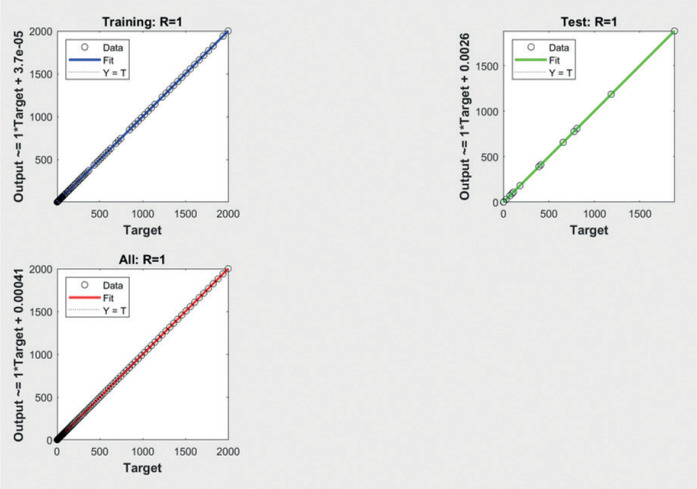


Figure 15: Correlations between targets values and output values in the training process, the testing process, and both processes

Example of neural net fitting function with measured values creation

This example aims to learn the neural network the values we obtained by measuring. We follow the methodology presented in section 2.1 of this chapter.

Step 1: Data preparation. We have 500 measured data values stored in the file data4 (see Figure 16). For clarity, we plot the x-axis from the value 0.1 to the value 50 with a step of 0.1. See the code below:

```
x=0.1:0.1:50
save data3 x
load data4 y
plot(x,y)
```

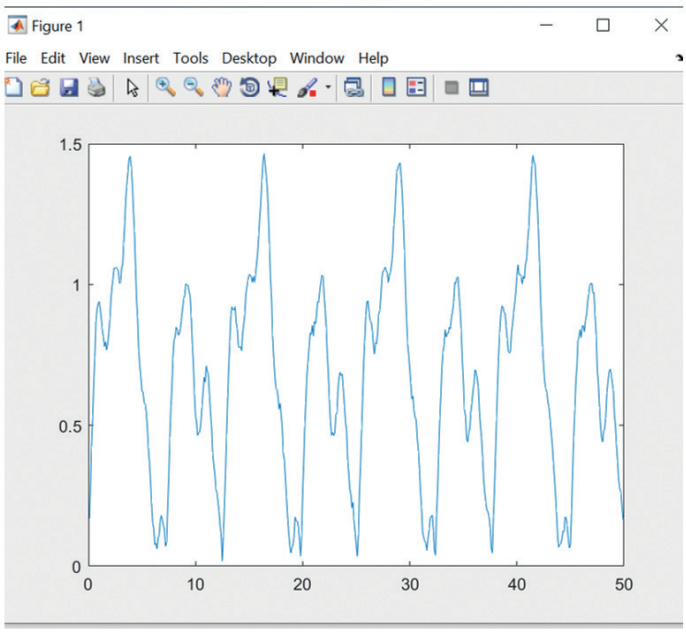


Figure 16: Measured values

Sometimes the measured data is noisy and therefore needs to be adjusted [1]. We can use a moving mean that smooths the data so the neural network can learn it. The moving mean gradually goes through all the smoothed values and replaces the current value with the average value. The window consists of the current value and a certain number of values before and after the current value. Now we will show how to smooth our measured data stored in the data4.mat file. We will extend our listing 2 with commands by smoothing the measured data using a moving mean with a window that is nine values wide. We will save the modified data in the vector m to the data5.mat file and use them as targets when learning the network (see code below) and Figure 17.

```
x=0.1:0.1:50
save data3 x
load data4 y
plot(x,y)
m = movmean(y,7)
plot(x,y,x,m)
save data5 m
```

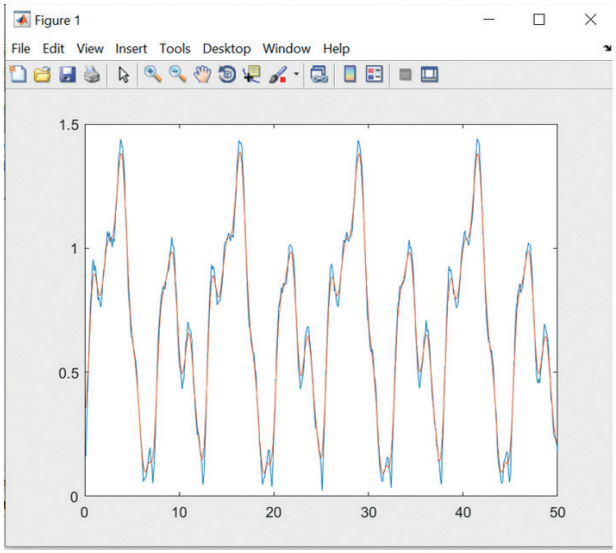


Figure 17. Measured values are in blue and smoothed data in red color

Step 2: We select the Neutral Net Fitting application in the Matlab environment.

Step 3: We load the data into the application. The file data3.mat contains the input data and the file data5.mat contains the targets.

Step 4: We leave the ratio as in the previous example.

Step 5: We will design the network architecture. We choose 50 neurons in the hidden layer.

Step 6: We choose the Bayesian Regularization learning algorithm.

Step 7: We start the learning process (see Figure 18).

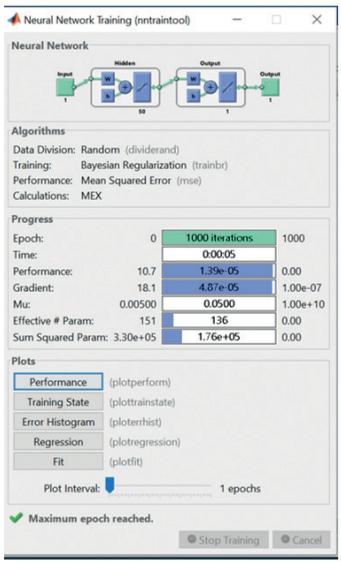


Figure 18: Network learning progress

Step 8: Let's take a closer look at the learning results of the network (Figure 19). We conclude that the network has learned the correct values based on the learning error of 1.39x10-5 and the network testing error of 1.39x10-5.

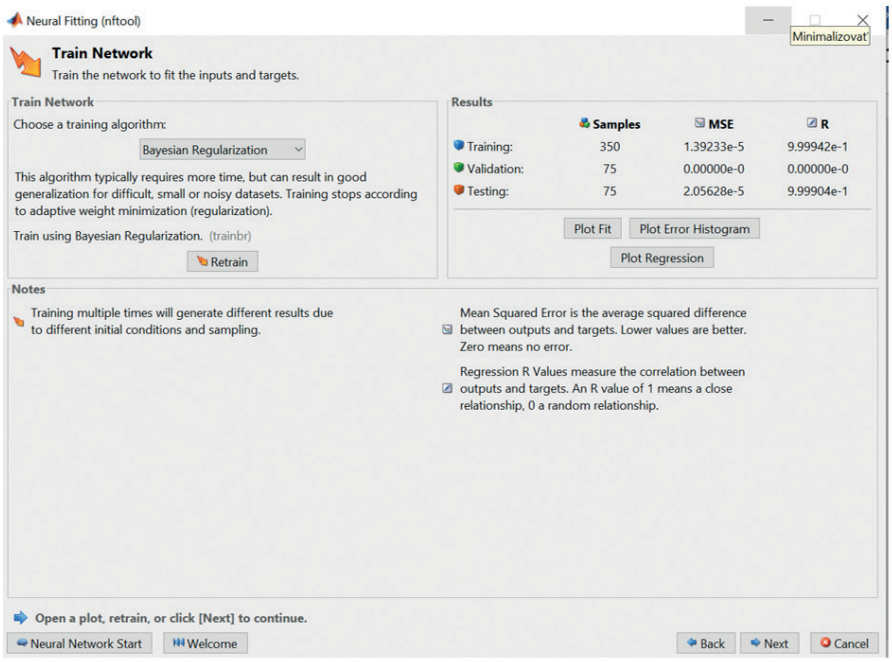


Figure 19: Display learning errors

In Figure 20, we can see the progress of the values of targets and outputs of the network depending on the inputs after learning and testing the network. The learned values overlap with the target values. In the lower part of the image, we see the displayed errors, which are minimal.

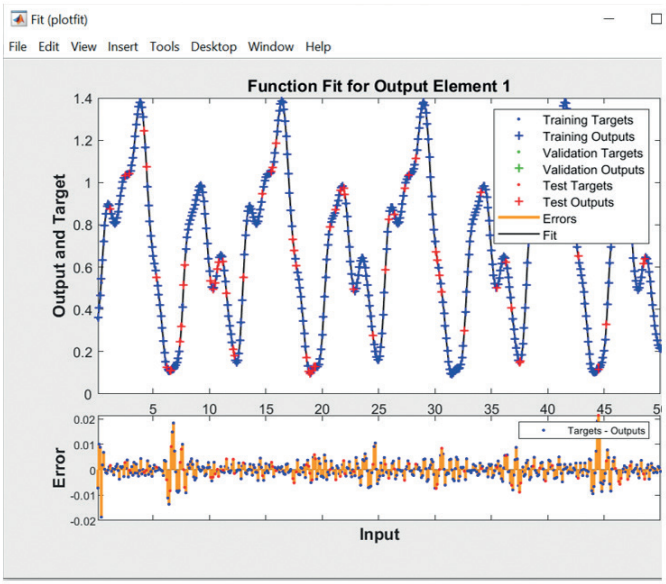


Figure 20: Network learning results - progress targets and learned values.

On the histogram, in Figure 21, we can see the size and frequency of the errors, which again proves that the errors are minimal.

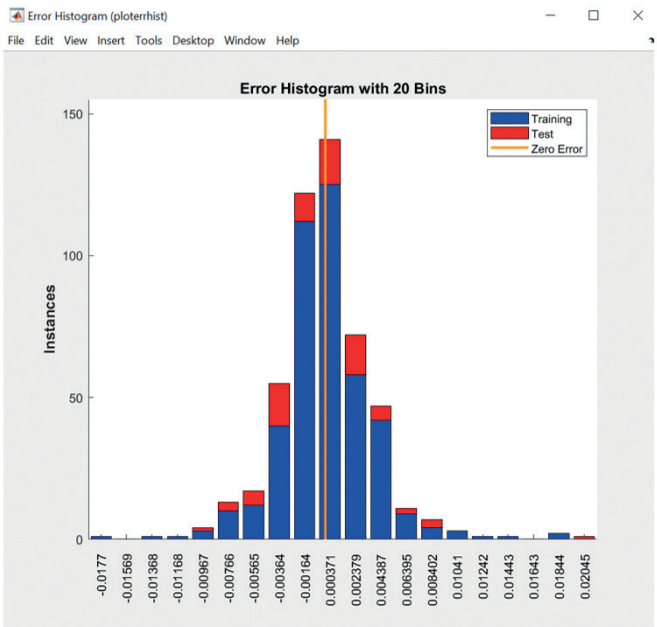


Figure 21: Results of network learning - histogram - size, and frequency of errors

The shown regressions (see Figure 22), which reached a value of 1, from the point of view of trained, tested, and all inputs equally show that we have a very well-trained network.

Overall, we can conclude that the achieved accuracy of learning and testing the network is sufficient, and the learning process of the network ends.

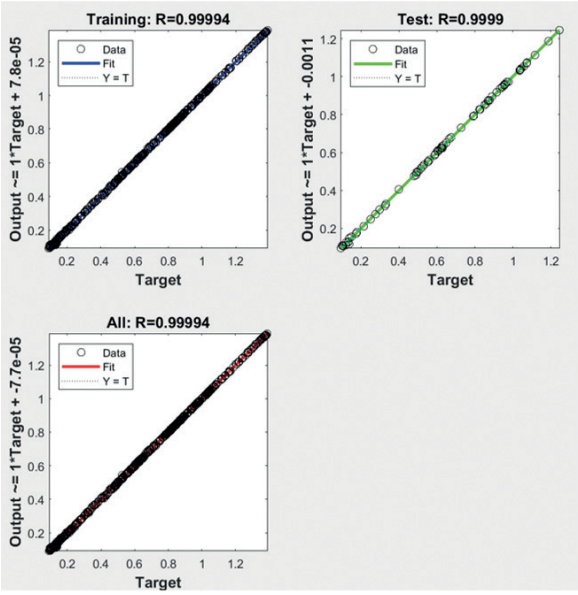


Figure 22: Network learning results - regression calculation

Example of data clustering by self-organizing map neural network

In this example, we solve the well-known task of classifying Iris flowers. We will use the IRIS dataset and describe a solved example that is part of the Matlab environment. Iris flowers can be described using 4 parameters, while the values (sepal length, sepat width, petal length and petal width) in the dataset are given in centimeters. Therefore, each flower is characterized by 4 elements. Our task is to create such a self-organizing map neural network that classifies iris flower types into classes such that similar types are located in a group close to each other. The map is created based on the similarity of the samples, and the learned neural network can classify even unknown samples [4].

We proceed according to the methodology of neural network implementation in Matlab graphical environment.

Step 1: We skip the step. We will use a ready-made dataset available in Matlab and describe this dataset in detail in step 3.

Step 2: In the Matlab environment, select the appropriate application from the APPS tab, from the Machine Learning category, select the Neutral Net Clustering application and launch the application. This application will help us create a self-organizing map neural network. See Figure 23.

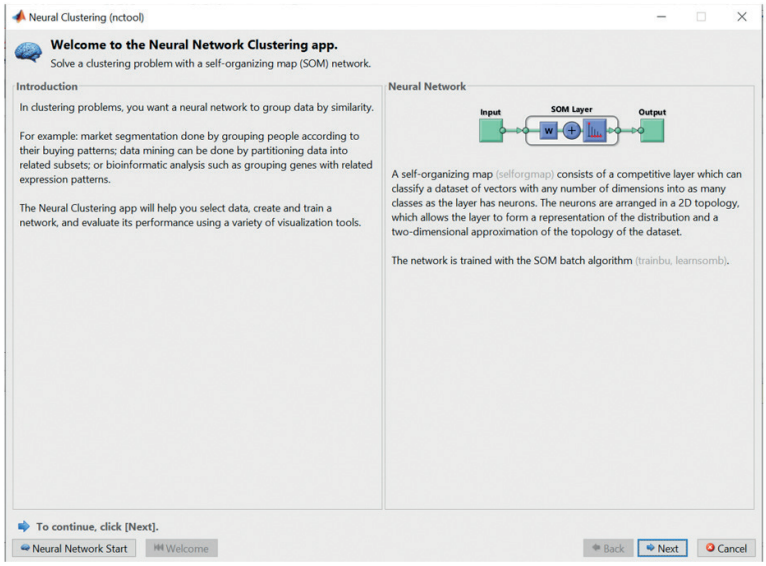


Figure 23: Application for Neural Network Clustering

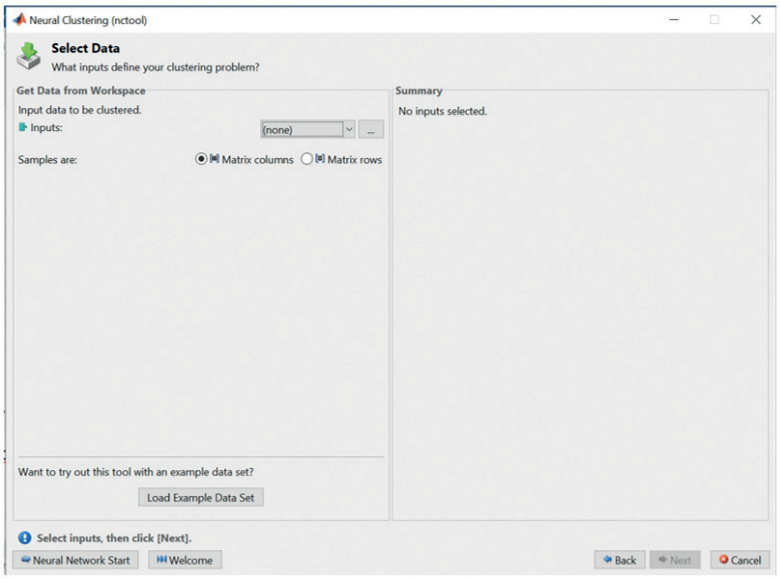


Figure 24: Loading prepared dataset

Step 3: We load the datasets. See Figure 24. Since we want to use the prepared IRIS dataset (see Appendix A), we press Load Example Data Set, select Iris Flowers, and import the data. See Figure 25.

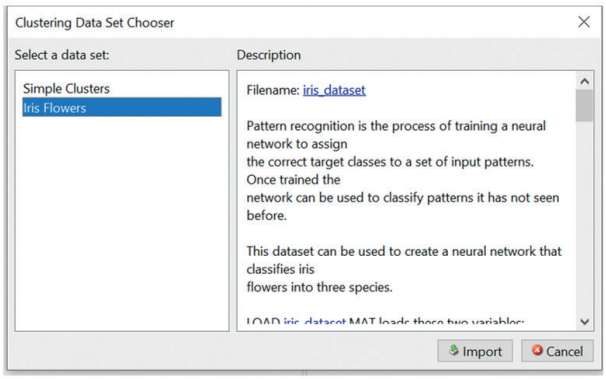


Figure 25: Import of Iris flowers dataset

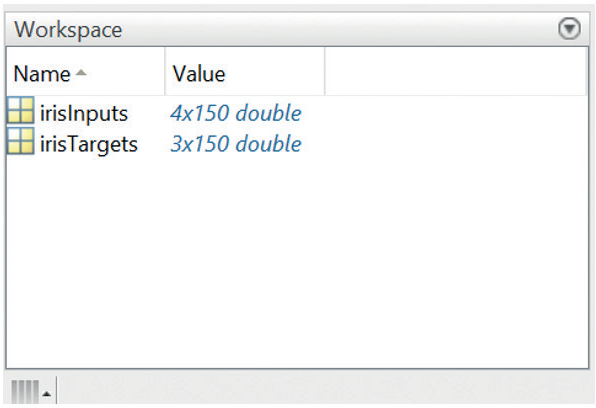


Figure 26: Matlab workspace after loading the Iris flowers dataset.

After loading the dataset, the irisInputs and irisTargets matrices are displayed in the Matlab workspace window, see Figure 26. We can see that the irisInputs matrix has dimensions of 4 rows x 150 columns. Four parameters describe one flower; therefore, one column represents one flower sample. The input dataset contains 150 flower samples. The irisTargets matrix specifies the classification of each input sample into one of 3 classes. We will write both matrices in the command window individually to better understand the data. Data samples can be seen in the codes below:

```
>> irisInputs
irisInputs =
    Columns 1 through 11
        5.1000 4.9000 4.7000 4.6000 5.0000 5.4000 4.6000 5.0000 4.4000 4.9000 5.4000
        3.5000 3.0000 3.2000 3.1000 3.6000 3.9000 3.4000 3.4000 2.9000 3.1000 3.7000
        1.4000 1.4000 1.3000 1.5000 1.4000 1.7000 1.4000 1.5000 1.4000 1.5000 1.5000
        0.2000 0.2000 0.2000 0.2000 0.2000 0.4000 0.3000 0.2000 0.2000 0.1000 0.2000
>> irisInputs
irisTargets =
```

Columns 1 through 18

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Columns 19 through 36

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Columns 37 through 54

1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Step 4: The division of samples into training and testing is preset, and therefore it is not necessary to enter it in this application.

Step 5: We will design the network architecture. In this case, the number of neurons of the SOM layer must be entered. A layer represents a two-dimensional square array. Therefore, when we specify an array size of 12, a two-dimensional array of 12x12 elements is created. See Figure 27. Then the map at the network output will be 12x12, i.e., 144 elements. The predefined output map topology is hexagonal.

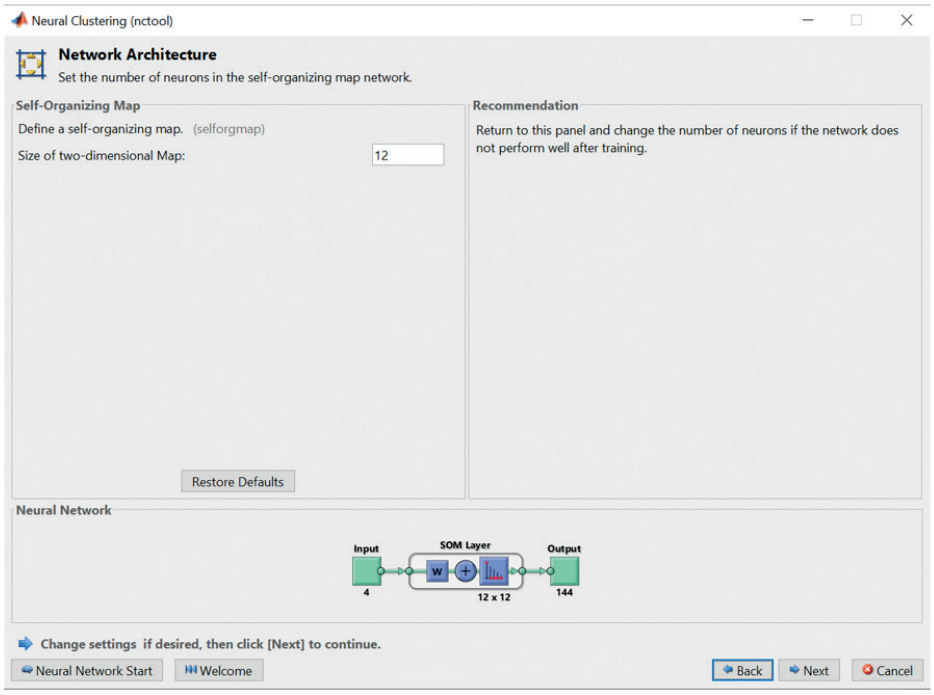


Figure 27: Network architecture design

Step 6: Selection of learning algorithm. The application has a predefined batch SOM algorithm, so we skip this step.

Step 7: We start the network learning process by pressing the Train button. The number of learning epochs is set to 200, and we can follow the learning process as shown in Figure 28.

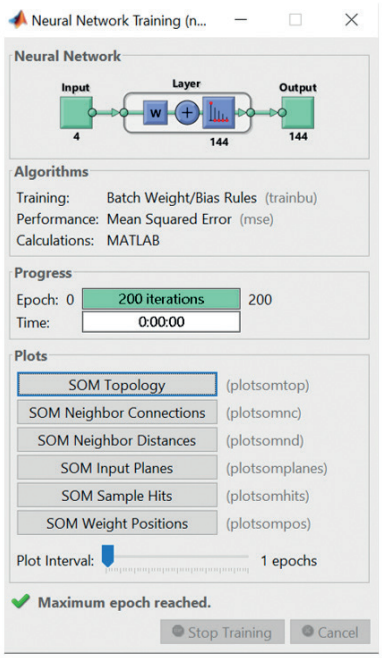


Figure 28: Network learning progress

Step 8: We display the results of learning the network using four buttons (see Figure 29). The first result is the classification of classes for each flower, and Plot som sample hits (see Figure 30) shows the number of flowers in each class. Areas of neurons with higher values represent classes of similarly frequently represented flowers. Conversely, areas with small values mean flowers with less abundance.

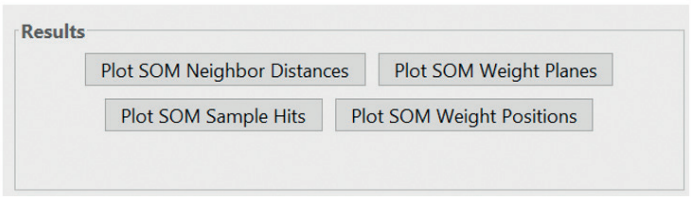


Figure 29: Network learning results

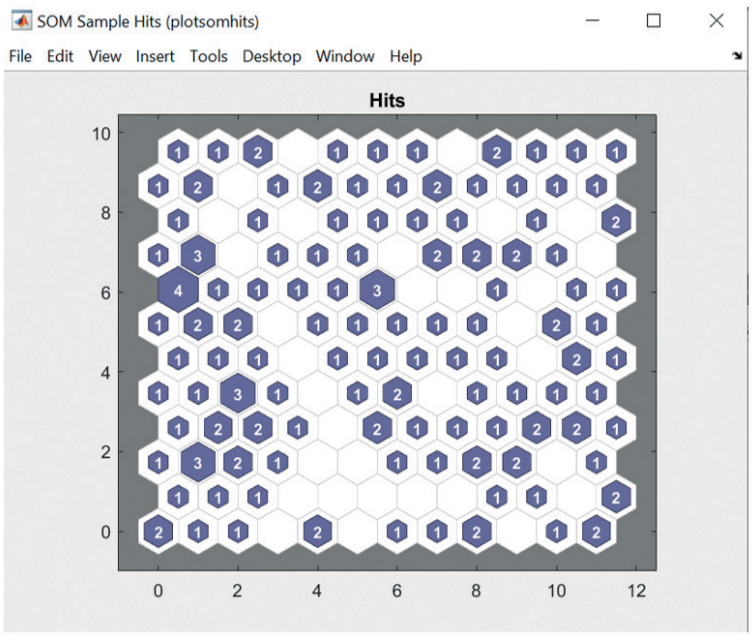


Figure 30: Network learning results - number of flowers in each class

The result that we display using Plot SOM Neighbor Distances expresses the Euclidean distance of a neuron class from its neighbors. Groups of neurons that form bright connections mean a high similarity of flowers in the input set. Conversely, dark-light connections represent cultivated areas with fewer flowers or areas without flowers. See Figure 31. Dark borders (joints) separate large areas of input space and indicate that flowers in separate areas have different features.

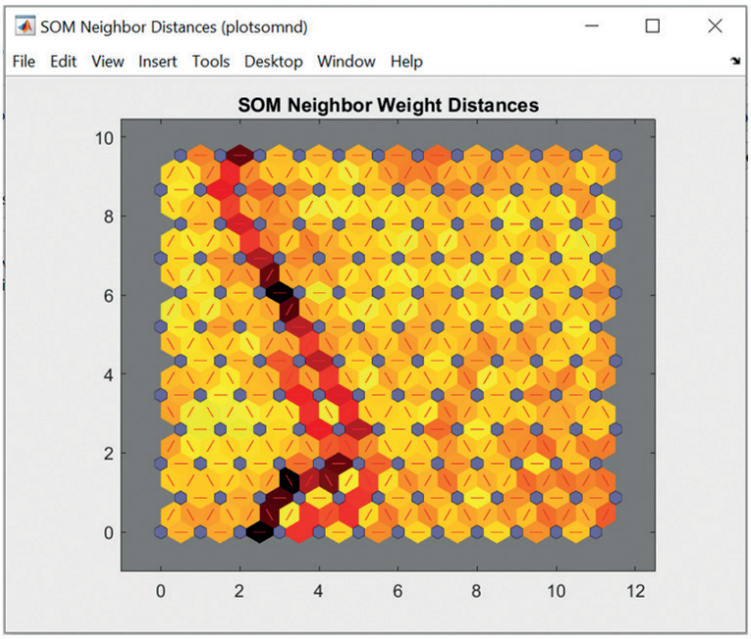


Figure 31: Results of network learning - abundance and classes of flowers in the input space

The resulting weights of the network from the point of view of the four input features of the flowers will be displayed using Plot SOM weight planes. See Figure 32.

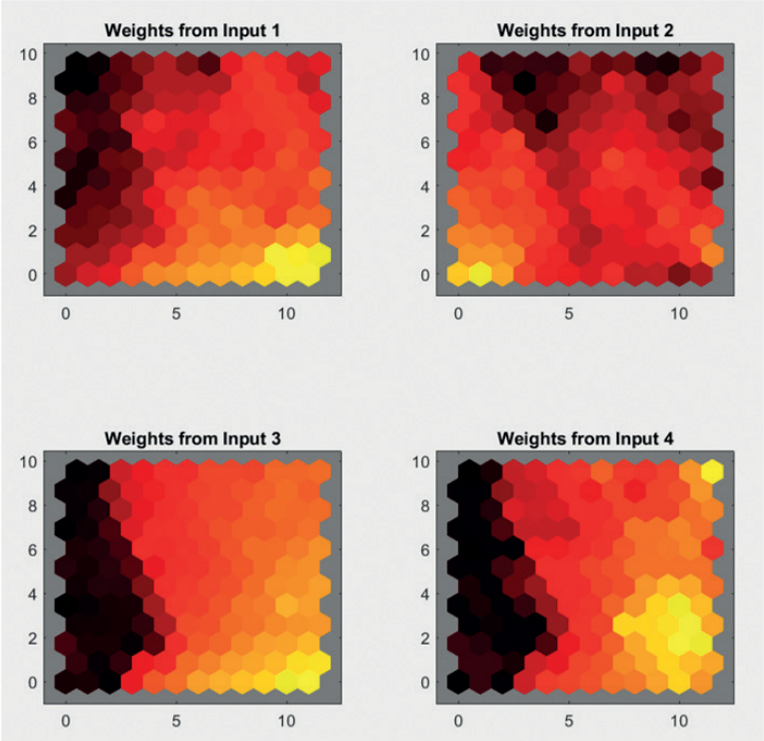


Figure 31: Network learning results - weight maps for individual network inputs

The weights connect each input to each of the network's 144 output neurons. Dark colors represent larger weights. Entries that have the same color on the map are strongly correlated.

CHAPTER 12

APPENDICES

This section contains appendices to the work presented in the main body of the handbook. Five appendices contain various data and exercise information related to the handbook and course in which the handbook can be used, specifically:

- **Appendix A** describes the Iris dataset used in the examples of sections 3 – 8.
- **Appendix B** contains examples of solutions to problems presented in section 7.
- **Appendix C** focuses on presenting selected air pollution and climate change datasets usable as sources for data analysis.
- **Appendix D** describes the impact of air pollution on human health.
- **Appendix E** contains proposed curriculum for course in which the handbook can be used.

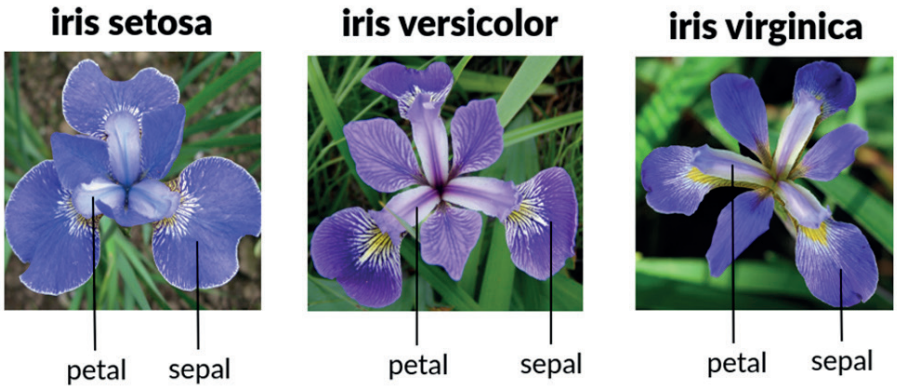
A-BRIEF DESCRIPTION OF THE IRIS DATASET

This part of the handbook was written by Alžbeta Michalíková and Adam Dudáš from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

The Iris dataset is one of the most frequently used datasets in data analysis, working with prediction models and tuning algorithms for data processing.

Edger Andersen created this dataset, first presented in the context of data analysis in the publication Fisher, R.A. *"The use of multiple measurements in taxonomic problems" Annual Eugenics, 1936.* The dataset comprises **five attributes** measured over **150 individuals** of the Iris flower (the dataset is 150×5 size). The flower consists of six leaves structured in two cycles where:

- the sepals form the inner cycle of the flower,
- the petals form the outer cycle of the flower.



The Iris dataset is compiled by measuring two values for each type of leaf (width and length), which creates **four numerical attributes**:

- sepal length and sepal width measured in centimeters or millimeters,
- petal length and petal width measured in centimeters or milimeters.

The fifth attribute of the dataset is the categorical value **class** or sometimes **species**, which divides the entities of the dataset into three classes:

- iris setosa,
- iris versicolor,
- iris virginica.

Each of these classes is represented in the dataset evenly - by **50 entities**. We present an example of one sample entity from each Iris dataset class:

Entity	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	setosa
2	7.0	3.2	4.7	1.7	versicolor
3	6.3	3.3	6.0	2.5	virginica

Working with the Iris dataset

The Iris dataset is so standardized that most data processing and analysis tools have an internal command that can be used to load this dataset.

For example, in the language R, instead of the name of the data file, we use just *iris*.

Example: By typing the name of the loaded dataset in R, we get console output containing all of the attributes and entities of the dataset. In the case of the iris dataset, we can type iris (without the need to load the dataset).

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
```

In the case of working with a tool that does not have an iris dataset available in this way, it is possible to freely download the dataset, for example, at:

https: //archive.ics.uci.edu/ml/datasets/iris

B-SOLUTIONS TO FUZZY CLASSIFICATION QUESTIONS

This part of the handbook was written by Alžbeta Michalíková from the Department of Computer Science, Faculty of Natural Sciences, Matej Bel University in Banská Bystrica, Slovakia.

Using the Sugeno method, classify data from the dataset Iris into a suitable number of classes.

Solution:

Answer the following questions:

1. How many **input variables** are in the Iris dataset?

   We have four input variables in the dataset IRIS.

2. What will we use **to describe input variables**?

   To describe input variables, we will use fuzzy membership functions.

3. What type of fuzzy **membership functions** will we use?

   We will use trapezoidal membership functions.

4. What will be the **output**?

   The output will be the specific class into which belong individual Iris flowers (rows of the table/ objects).

5. What will we use **to describe the output variables**?

   To describe the output variables, we will use constant functions (constants).

6. Which type of rules will we use?

   We will use Sugeno IF-THEN rules.

7. Write an example of one rule!

   If Input1 is small and Input2 is small, Input3 is middle, and Input4 is high, then Output is class1 (or Iris_Setosa).

Determine the values of input variables parameters from these data and fill them into the following tables.

Table B.1: Parameters of input variables

INPUT 1:

Name	Parameter
Universe	[40 80]
Red	[-20 -10 48 59]
Blue	[48 55 67 71]
Green	[55 71 81 90]

INPUT 2:

Name	Parameter
Universe	[20 45]
Red	[22 39 46 50]
Blue	[0 10 24 35]
Green	[21 28 34 39]

INPUT 3:

Name	Parameter
Universe	[10 70]
Red	[0 5 19 28]
Blue	[26 30 44 52]
Green	[44 53 75 80]

INPUT 4:

Name	Parameter
Universe	[0 25]
Red	[-10 -5 6 10]
Blue	[6 10 13 19]
Green	[13 19 30 35]

Determine the values of output parameters. Fill the following table with the correct values, if for the **output linguistic variable**, we use **constant functions**.

Table B.2: Parameters of output variables

OUTPUT:	
Name	**Parameter**
Universe	**[1 3]**
Red	1
Blue	2
Green	3

Design the number of rules and write them in the correct form.

Rules:

1. If Input1 is Red and Input2 is Red and Input3 is Red and Input4 is Red then Output is Red.
2. If Input1 is Blue and Input2 is Blue and Input3 is Blue and Input4 is Blue then Output is Blue.
3. If Input1 is Green and Input2 is Green and Input3 is Green and Input4 is Green then Output is Green.

C-BRIEF DESCRIPTION OF CLIMATE CHANGE DATASETS

This part of the handbook was written by Mihaela Tinca Udristioiu from the Department of Physics, Faculty of Sciences, and Silvia Puiu, from the Department of Management, Marketing and Business Administration, Faculty of Economics and Business Administration, University of Craiova, Romania.

Research can progress easier and faster when researchers and users generally have open-source access to information. With the Internet at our fingertips, we need to know where to look for accurate, up-to-date, and reliable sources of information. All these reasons emphasize why the role of databases is so important. The data is structured and usually in a way that can be easily transformed and processed by the needs of the researcher or user.

The European Commission has created a few open-source data about climate change. It is easy to download data sets; it is more difficult to process and analyze (exploratory and predictive analysis) datasets and use different algorithms to create mathematical models. Copernicus, European Climate Assessment and Dataset, Climate Explorer, and Indecis are only a few. We need data to monitor climate change and forecast weather, observe climate sensitivity to various parameters, create different scenarios, and see the evolution of some processes in the short and long terms. In the following, the authors will concisely present some databases.

***European Center for Medium-range Weather Forecast (ECMWF)** processes data from around 90 satellite instruments in daily operational data assimilation and monitoring activities. About 60 million quality-controlled observations are available daily in the Integrated Forecasting System, most of which are satellite measurements. ECMWF also benefits from all observations from non-satellite sources, including surface-based and aircraft reports.*

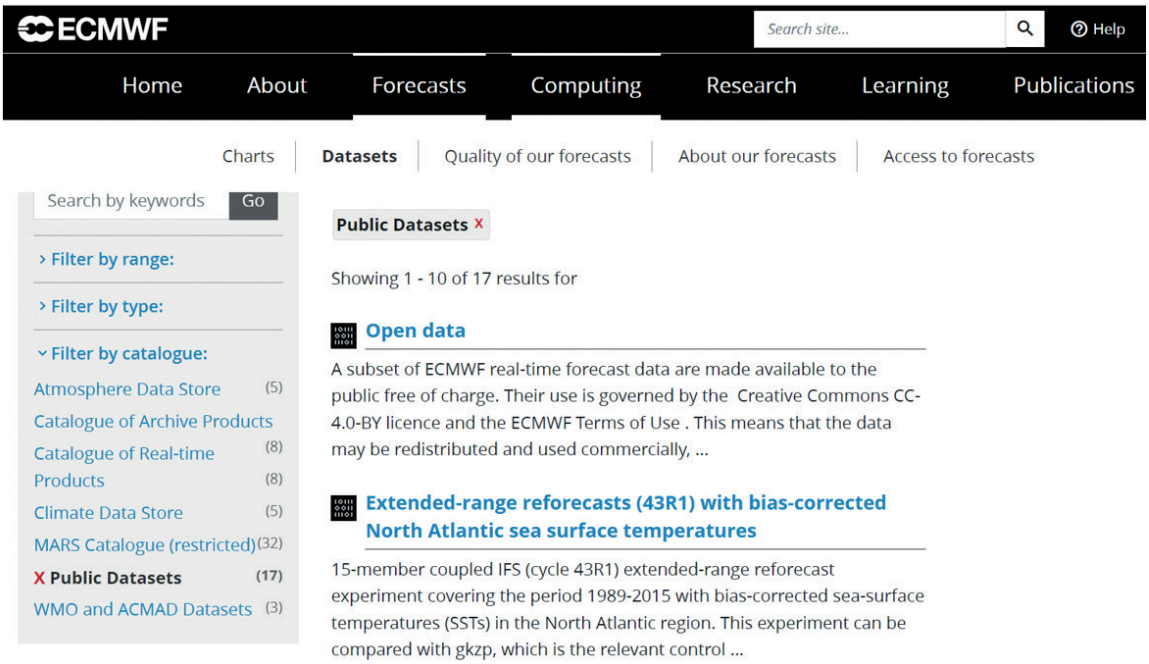


Figure C.1: Printscreen of the Public Data Set section from ECMWF (source: https://www.ecmwf.int/en/forecasts/datasets/search)

Copernicus is the Earth observation component of the EU Space Program. The European Commission (EC) manages it. EC implements Copernicus in partnership with the EU Member States, the European Space Agency (ESA), the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT), the European Centre for Medium-Range Weather Forecasts (ECMWF), the Joint Research Centre (JRC), the European Environment Agency (EEA), the European Maritime Safety Agency (EMSA), Frontex, SatCen and Mercator Océan. Copernicus contains climate data sets from various sources (reanalyzes, satellite products, climate projections). The Copernicus database is one of climate change's most frequently used datasets, working with prediction models and tuning algorithms for data processing. It has satellites (SENTINELS 1-6) with prominent missions.

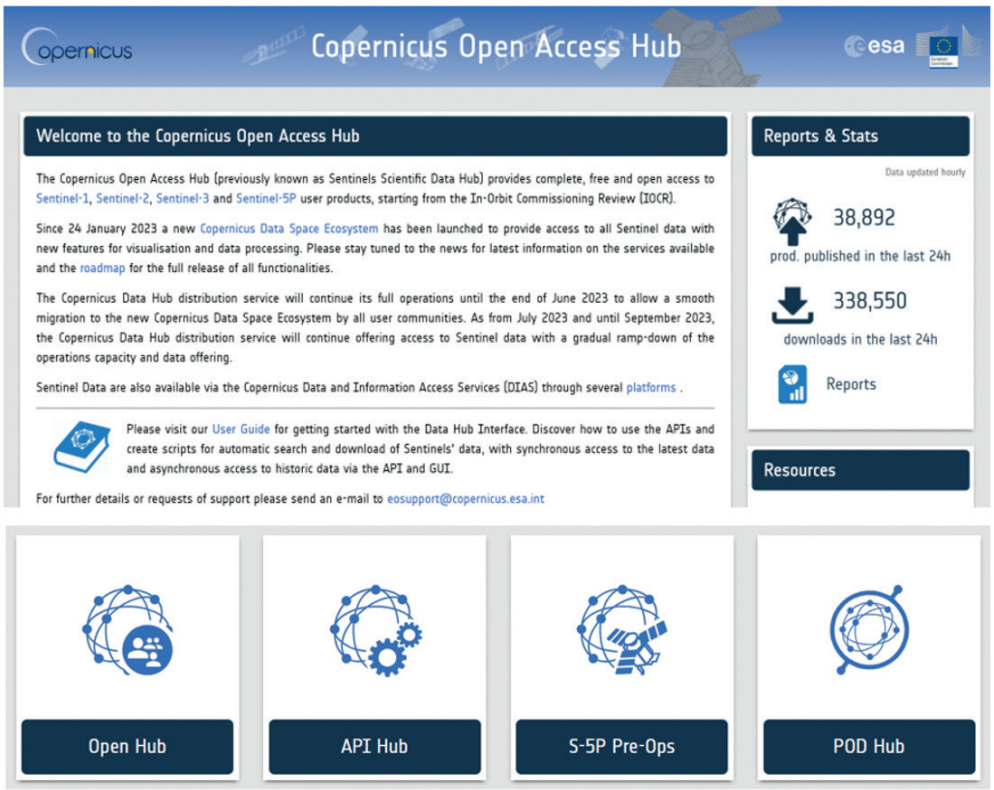


Figure C.2: Image of the interface of Copernicus Open Access Hub (source: https://scihub.copernicus.eu/)

The SENTINEL-1 has two polar-orbiting satellites and operates 24 hours from 24 and 7 days from 7, without holidays. It uses radar imaging to gather images regardless of the weather.

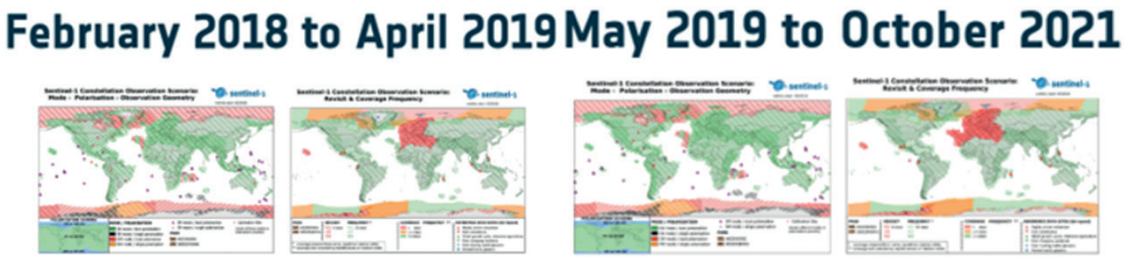


Figure C.3: Image provided by the SENTINEL 1 for two-time intervals
(source: https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-1/observation-scenario/archive)

SENTINEL-2 consists of two polar-orbiting satellites in the same sun-synchronous orbit, 180° apart. They keep track of changes in land surface conditions. Its large sweep width (290 km) and long revisit time (10 days at the equator with one satellite and five days with two satellites in cloud-free circumstances, resulting in 2-3 days at mid-latitudes) will aid in monitoring changes to the Earth's surface. SENTINEL-2 products are available for users. Some products are only for experts (top-of-atmosphere radiances in sensors geometry), and others are for all users (top-of-atmosphere reflectances in cartographic geometry and atmospherically corrected surface reflectances in the same geometry). The pilot products are generated only on demand. There are two categories of pilot products: Harmonized SENTINEL-2+Landsat-8/9 surface reflectances in cartographic geometry.

SENTINEL-3 takes measurements of sea-surface topography, sea and land surface temperature, and ocean and land surface color. The idea is to support ocean forecasting systems and environmental and climate monitoring.

SENTINEL-4 monitors key air quality trace gases and aerosols over Europe, supporting the Copernicus Atmosphere Monitoring Service (CAMS) with a fast revisit time. Spectrally and radiometrically calibrated and geo-located Earth radiance and spectrally and radiometrically calibrated solar irradiance are available as parameters for all users but data processing parameters, calibration, and instrument diagnostic data are only for expert users.

SENTINEL-5 is a high-resolution spectrometer system operating in the ultraviolet to the shortwave infrared range with seven different spectral bands: UV-1 (270-300nm), UV-2 (300-370nm), VIS (370-500nm), NIR-1 (685-710nm), NIR-2 (745-773nm), SWIR-1 (1590-1675nm) and SWIR-3 (2305-2385nm). Sentinel-5 provides information on air quality and composition-climate interaction (O_3, NO_2, SO_2, HCHO, CHOCHO, and aerosols). Sentinel-5 delivers quality parameters for CO, CH_4, and stratospheric O_3 with daily global coverage for climate, air quality, and ozone/surface UV applications.

SENTINEL-5P performs atmospheric measurements with a high spatiotemporal resolution for air quality, ozone & UV radiation, and climate monitoring & forecasting.

Copernicus SENTINEL-6 Michael Freilich is focused on sea level rising because of climate change and is the next radar altimetry reference mission to extend the legacy of sea-surface height measurements until at least 2030.

Another important database is **ECA&D** which contains observations from weather stations and data sets derived from them at the European level; the researchers consider these datasets reference data. This site contains information regarding changes in weather and climate extremes and the daily dataset needed to monitor and analyze these extremes.

Figure C.4: The interface of ECA&D and WMO (source: https://www.ecad.eu)

KNMI Climate Explorer is another database that contains a class of climate data (time series or field) from reanalyzes and climate models, including climate projections; it has the advantage of a friendlier interface (including graphical representations). For this reason, it represents a good educational tool. Users can download a time series about daily and monthly station data and climate indices. At the annual level, only annual climate indices are available. The researchers can download this information by selecting a field like daily fields, monthly observations, monthly reanalysis fields, monthly and seasonal historical reconstructions, monthly seasonal hindcasts, monthly CMIP3+ scenario runs, monthly CMIP5 scenario runs, annual CMIP5 extremes, monthly CMIP6 scenario runs, monthly CORDEX scenario runs, attribution runs.

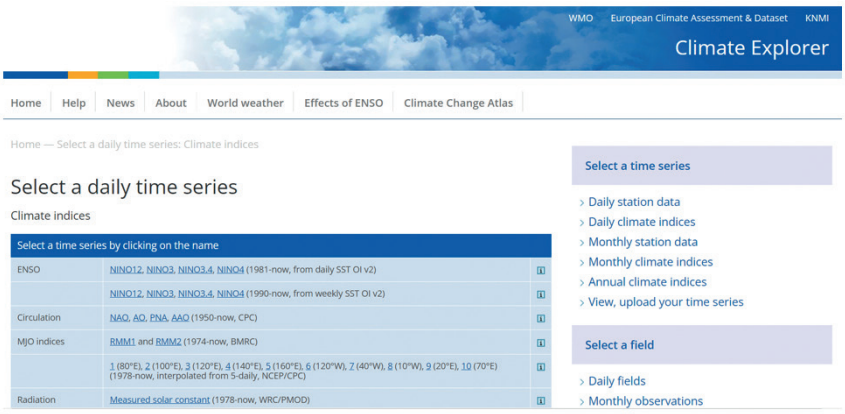


Figure C.5: Print screen of KNMI Climate Explorer
(source: https://climexp.knmi.nl/selectdailyindex.cgi?id=someone@somewhere)

Met Office Hadley Centre provides datasets of meteorological variables. The researchers use this information in climate monitoring and climate research. The classes are the following: key climate indicators, marine datasets, daily & hourly data/extreme indices, land surface data, combined land/marine pressure data, upper air data, one-off data accompanying journal articles, and older datasets.

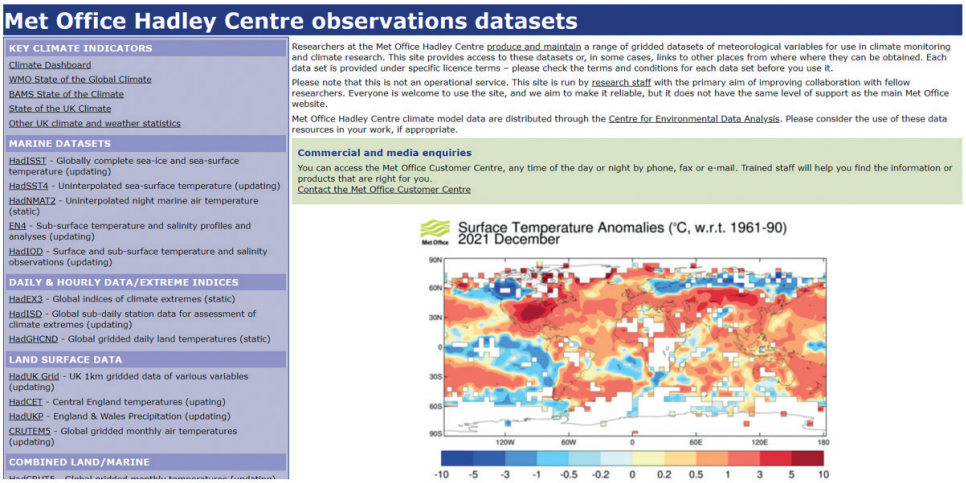


Figure C.6: The interface of the Met Office Hadley Centre (source: https://www.metoffice.gov.uk/hadobs/index.html)

Indecis contains climate data on agriculture, disaster risk reduction, energy, health, water, and tourism (*http://indecis.eu/indices.php*). Here there are only climatic indices - many, with various applications; the platform has definitions for climatic indices, with graphic representation as a map and a series of data in a single point, plus download. This database is also a good educational tool. It contains daily station data, quality-controlled station data, homogenized station data, recovered station data, and gridded versions of the indices.

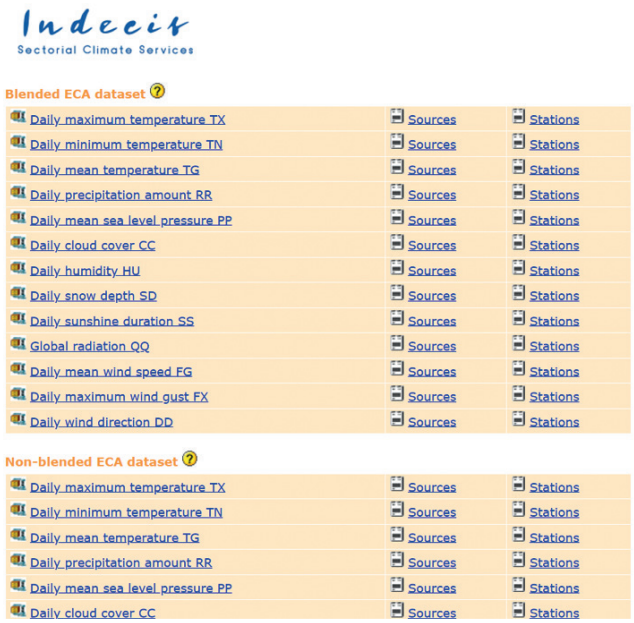


Figure C.7 : Classes of data that can be downloaded from Indecis (source: https://www.ecad.eu/dailydata/predefinedseries.php)

There are open-source data from the European Environment Agency for air quality data.

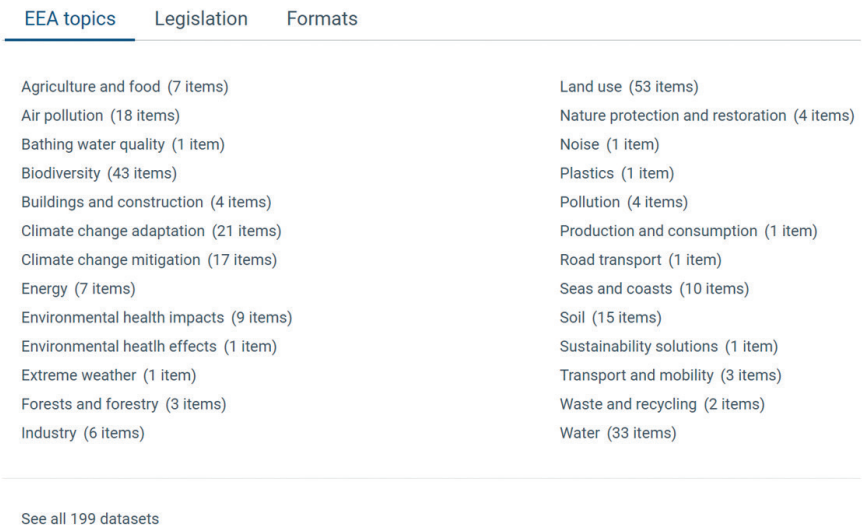


Figure C.8: Datasets given by the European Environment Agency
(source: https://www.eea.europa.eu/themes/air/explore-air-pollution-data)

World's Air Pollution contains sensors from the National Environment Agencies and gives information about real-time air quality index.

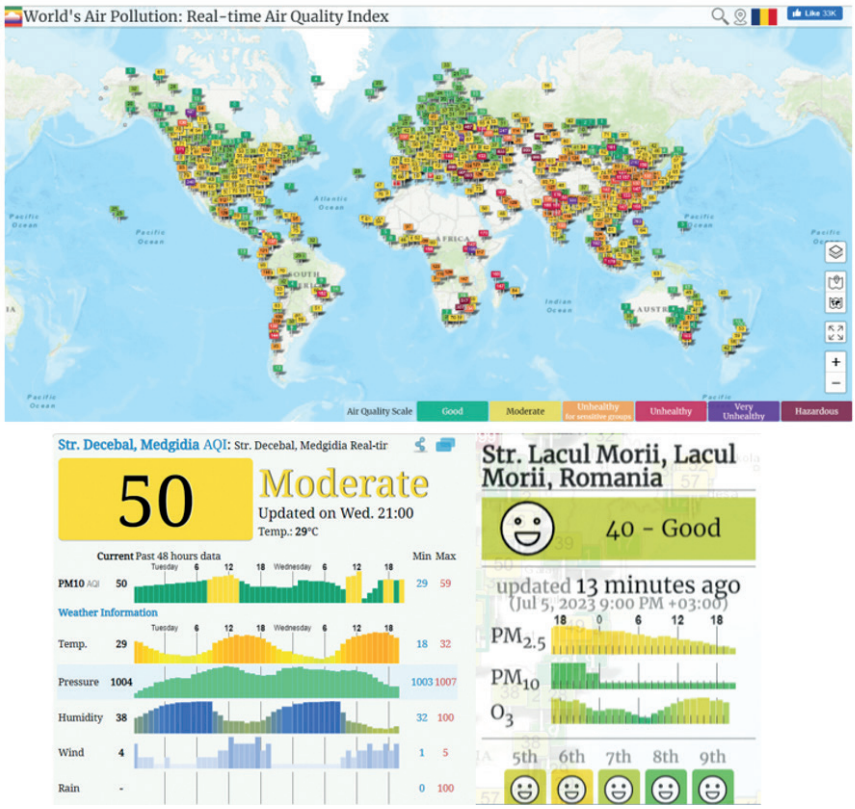


Figure C.9 : The map of air quality sensors worldwide and more information when clicking on a sensor
(source: https://waqi.info/)

OECD contains information about indicators like air and GHC emissions, air pollution exposure, and air pollution effects as charts, maps, or tables to be simple for everyone to visualize the evolution in time of the data. A simple click will open data organized in charts, maps, and tables.

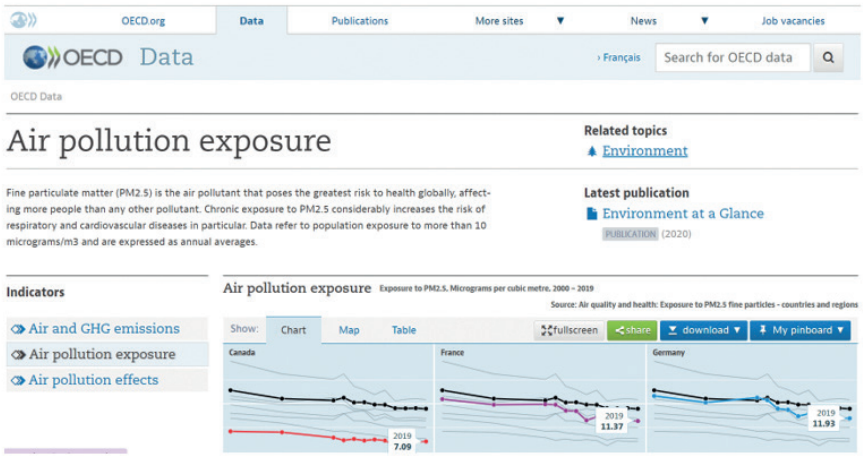


Figure C.10 : OECD interface (source: https://data.oecd.org/air/air-pollution-exposure.htm)

There are also citizen science initiatives and community-driven sensor networks. These networks contain low-cost sensors that monitor air quality by citizens in their communities and have excellent coverage of a large area of Europe. Some of these networks have been built by volunteers in some projects for educational purposes. uRADMonitor® is such an example in Romania. The network provides open access to real-time data. The administrators can provide historical data on request. Citizen science initiatives promote transparency and accountability in environmental monitoring. Other examples are the following: Community Air Sensor Network (CAIRSENSE), the Smart Citizen® network, the Public Laboratory for Open Technology and Science, or the Public Lab network, the Eye on Earth initiative, the Global Learning and Observations to Benefit the Environment (GLOBE), the HabitatMap©, the Imperial County Community Air Monitoring Project, and the Citizen Weather Observer Program (CWOP).

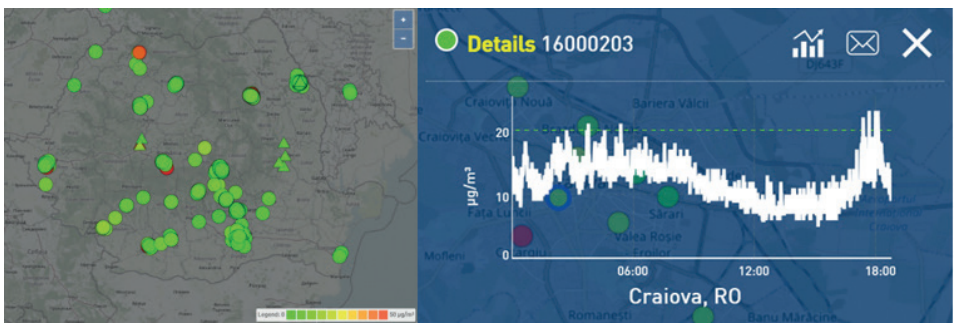


Figure C.11: Printscreen uRADMonitor® network (source: https://www.uradmonitor.com/)

Statistical databases help research advance, providing important data for multiple variables and longer periods. This information is fundamental in drawing conclusions, creating, predicting, or mitigating scenarios. For example, if we need data on climate change, such as pollution levels, we have

multiple open-source databases to access and use for our objectives. Decisions are based on data inputs; our choices are only good if we have the correct information. So, it is important to choose reliable sources of information, such as those from well-known national and international organizations.

One of these sources is the Eurostat database, which provides statistical data about many aspects of interest for European countries. One of the reasons we can be sure about this database is its long history (70 years) and the fact that it is under the umbrella of the European Union.

Let us take an example of how we can access data on climate change using the Eurostat database. We can access the website directly and search climate change or do that using a search engine. If we go to *https://ec.europa.eu/eurostat/web/climate-change/database*, we can find multiple data for our aim, as shown in Fig. C.12.

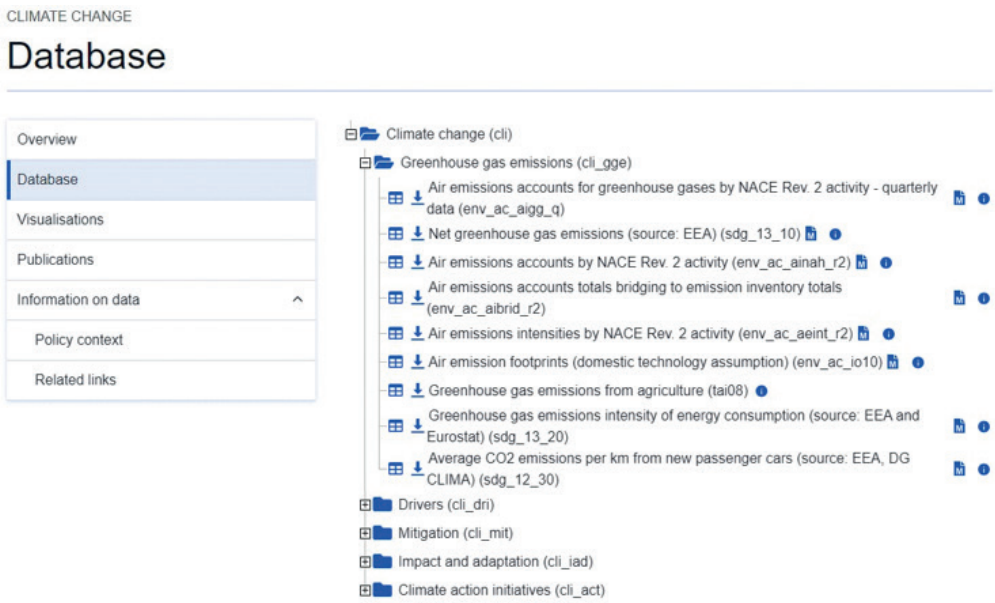


Figure C.12: Print screen from the Eurostat website regarding the information on climate change (source: https://ec.europa.eu/eurostat/web/climate-change/database)

The database on climate change has many folders: greenhouse gas emissions; drivers of climate change; mitigation of climate change; impact and adaptation; and climate action initiatives. Each one of them has data that the user can download. Information is free and open to everyone in several formats.

Under the greenhouse gas emissions folder, we can notice there are several pieces of data. If we go to the first one - Air emissions account for greenhouse gases (quarterly data), we can find more information if we click the right button. The window displayed is the one in Figure C.13. Thus, we can see that the data is available for 13 years, from 2010 to 2022. The database was updated in May 2023.

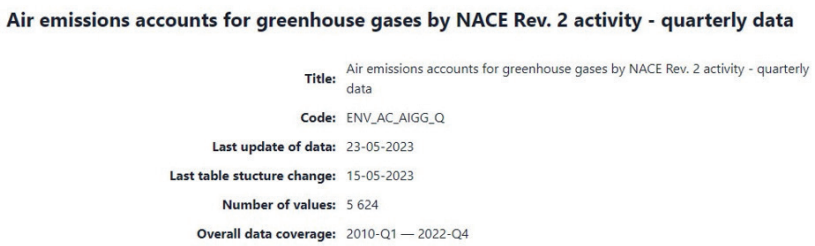


Figure C.13: Print screen after clicking the information button (source: https://ec.europa.eu/eurostat/web/climate-change/database)

If we are interested to find out more about the drivers of climate change, we select the second folder, and we notice in Figure C.14 that there are data for all important drivers, such as energy; transport; industrial processes; waste; agriculture; and land use, land use change and forestry. For Energy, we can download data regarding final energy consumption, final energy consumption per capita, final energy consumption by sector, etc.

Selecting the data the user needs to reach his objectives is important. Data is raw and unprocessed, so the user can use several tools to process the data and notice a trend, predict some scenarios, and provide the results he came up with. Businesses, individuals, governments, and others with responsibilities will use these results to prevent or improve some aspects.

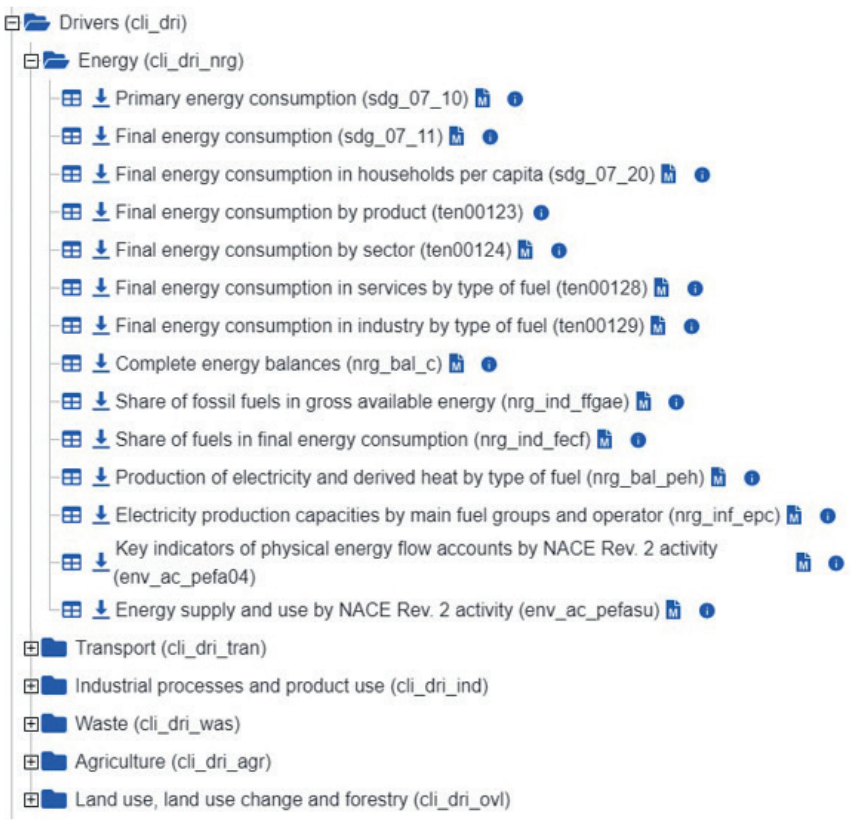


Figure C.14: Print screen on the information from Eurostat regarding the drivers of climate change (source: https://ec.europa.eu/eurostat/web/climate-change/database)

Now, let us see how the information looks if we want to check the final energy consumption in households per capita. Fig. C.12 shows a code between brackets near this indicator: SDG 7. This information is, in fact, a reference to the seventh sustainable development goal from the 2030 Agenda of the United Nations. This refers to Affordable and Clean Energy.

If we click on the first icon that looks like a table, we can read explanations regarding the indicator, but we also can choose the data format (table, line, bar, map) and the variables we need (countries and years) – Fig. C.15 and C.16.

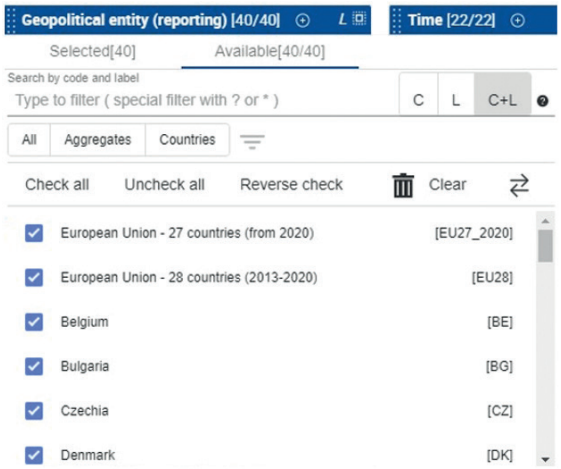


Figure C.15: Print screen on the filters that can be used for the data from Eurostat (source: https://ec.europa.eu/eurostat/web/climate-change/database)

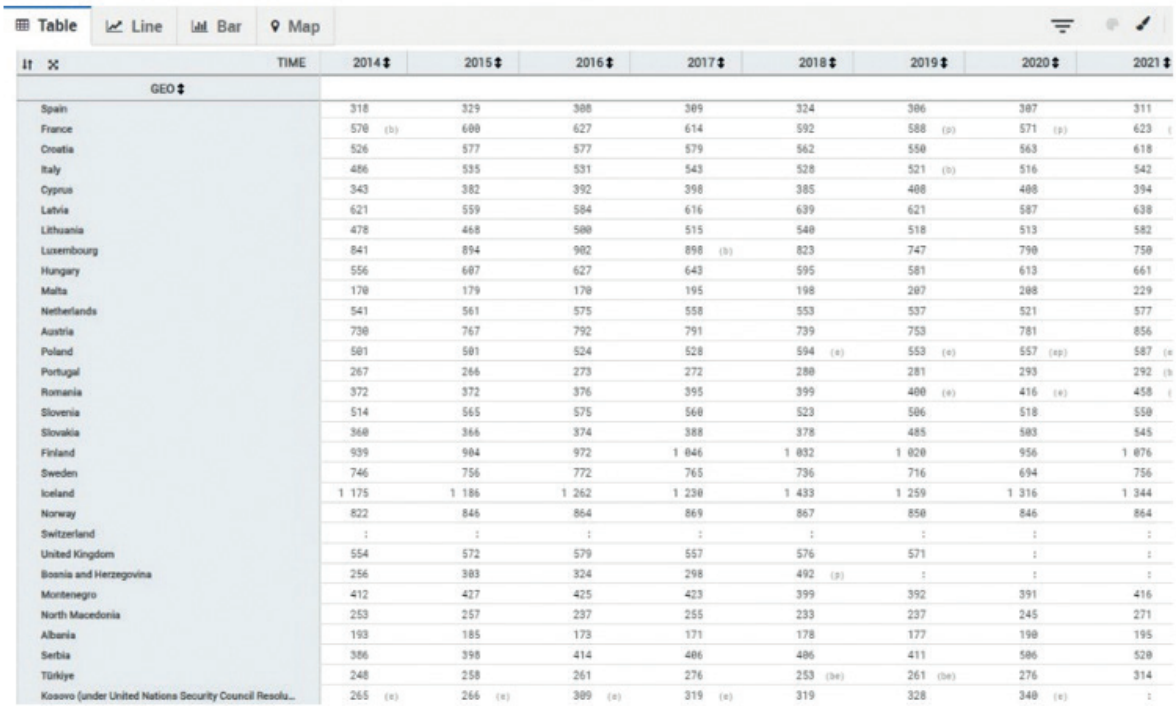


Figure C.16 : Print screen on the information displayed if we choose the table format (source: https://ec.europa.eu/eurostat/web/climate-change/database)

Besides Eurostat, the researchers can use other open-source databases. We can mention Our World in Data which has as a main objective stated on their website (*https://ourworldindata.org/*): they publish 'research and data to make progress against the world's largest problems, which refer to the population dynamic, energy and environment, health, food, poverty, education, living conditions, human rights, technological changes, and violence and war. It is under the umbrella of a non-profit but is highly cited in the literature review and the media.

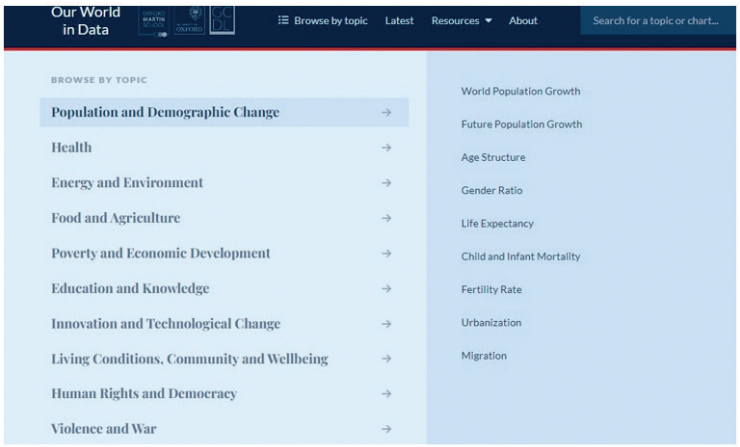


Figure C.17: Print screen from Our World in Data regarding the topics addressed by this publication (source: https://ourworldindata.org/)

If we are interested in air pollution, we select Energy and Environment and can choose outdoor or indoor air pollution. This website offers articles and raw statistical data for your research or activity. Thus, you can discover how many deaths globally can be attributed to air pollution (Fig. C.18). You can also find the outdoor air pollution rate by age and download the data as a table or chart (Fig. C.19).

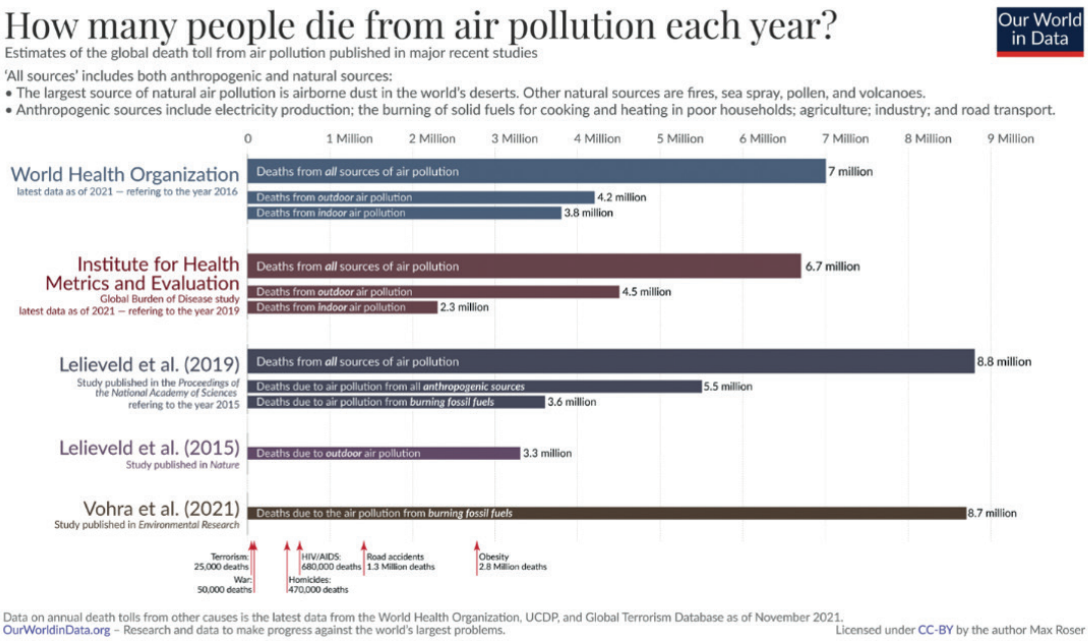


Figure C.18: Global deaths from air pollution (source: https://ourworldindata.org/data-review-air-pollution-deaths)

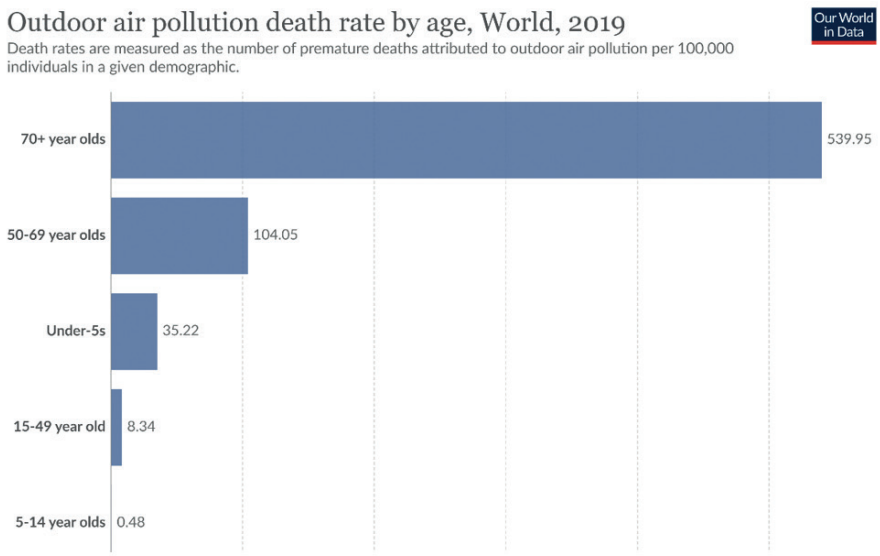


Figure C.19: Air pollution death rate by age
(source: https://ourworldindata.org/outdoor-air-pollution#outdoor-air-pollution-deaths-by-age)

D-IMPACT OF AIR POLLUTION ON HUMAN HEALTH

This part of the handbook was written by Slaveya Petrova from the Department of Ecology and Environmental Conservation, Faculty of Biology, Plovdiv University "Paisii Hilendarski," Bulgaria.

Air pollution is the contamination of the indoor or outdoor environment by any chemical, physical, or biological agent that modifies the natural characteristics of the atmosphere.

Household combustion devices, motor vehicles, and industrial facilities are common sources of air pollution. Pollutants of major public health concern include particulate matter (PM), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), and sulfur dioxide (SO2).

According to the European Environment Agency (EEA), each one of the air pollutants could be associated with a different source/s:

- Residential, commercial, and institutional energy consumption was the principal source of particulate matter in 2020. The manufacturing and extraction industry and agriculture were also significant sources of PM10. A descending trend was observed between 2005 and 2020 about the emissions of particulate matter (PM10 and PM2.5) – they fell by 30% and 32%, respectively.
- Agriculture was the principal source of ammonia (94% of total emissions) and methane (56%) in 2020. Ammonia emissions fell by only 8% from 2005 to 2020. This was the lowest percentage reduction of all pollutants.
- Road transport was the principal source of nitrogen oxides in 2020, revealing 37% of emissions. A significant decline of up to 48% in the emissions of nitrogen oxides was found between 2005 and 2020.
- Energy supply sector was the principal source of sulfur dioxide, responsible for 41% of emissions in 2020. Emissions of sulfur dioxide fell by 79% between 2005 and 2020.
- Manufacturing and extraction industries and the energy supply sector were the principal sources of heavy metals emissions in 2020. Between 2005 and 2020, the largest reductions in emissions were found for nickel (64%) and arsenic (62%).

D.1 Types of pollutants and health risks

Particulate matter (PM)

Particulate matter (PM) is a common proxy indicator for air pollution. The major components of PM fractions are sulfates, nitrates, ammonia, sodium chloride, black carbon, mineral dust, and water.

The health risks associated with particulate matter of less than 10 and 2.5 microns in diameter (PM_{10} and $PM_{2.5}$) are especially well documented. PM can penetrate deep into the lung and enter the bloodstream causing cardiovascular (ischaemic heart disease), cerebrovascular (stroke), and respiratory impacts. Both long-term and short-term exposure to PM is associated with morbidity and mortality from cardiovascular and respiratory diseases. Long-term exposure has been linked to adverse perinatal outcomes and lung cancer.

Carbon monoxide (CO)

Carbon monoxide is a colorless, odorless, and tasteless toxic gas produced by the incomplete combustion of carbonaceous fuels such as wood, petrol, charcoal, natural gas, and kerosene. Carbon monoxide diffuses across the lung tissues and into the bloodstream, making it difficult for the body's cells to bind to oxygen. This lack of oxygen damages tissues and cells. Exposure to carbon monoxide can cause difficulties breathing, exhaustion, dizziness, and other flu-like symptoms. Exposure to high levels of carbon monoxide can be deadly.

Ozone (O_3)

Ozone at ground level is one of the major constituents of photochemical smog. It appears through the reaction with gases in the presence of sunlight. It is worth mentioning that ozone can be generated by household equipment, such as portable air cleaners. Exposure to excessive ozone can cause problems such as breathing, trigger asthma, reduce lung function, and lead to lung disease.

Nitrogen dioxide (NO_2)

NO_2 is a gas commonly released from fuel combustion in the transportation and industrial sectors. Household sources of nitrogen oxides (NO_x) include equipment that burns fuels, such as furnaces, fireplaces, gas stoves, and ovens. Exposure to nitrogen dioxide can irritate airways and aggravate respiratory diseases.

Sulfur dioxide (SO_2)

SO_2 is a colorless gas with a sharp odor, produced from burning fossil fuels (coal and oil) and smelting mineral ores containing sulfur. Exposure to SO_2 is associated with asthma hospital admissions and emergency room visits.

Polycyclic aromatic hydrocarbons (PAH)

Polycyclic aromatic hydrocarbons (PAH) are present in the atmosphere in particulate form. They are a group of chemicals formed primarily from incomplete combustion of organic matter (e.g., cooking of meat) and fossil fuels in coke ovens, diesel engines, and wood-burning stoves. Also, they can be present in tobacco smoke. Short-term exposure can irritate eyes and breathing passages. Long-term exposure to PAH has been linked to lung cancer.

D.2 WHO global air quality guidelines

Since 1987, WHO has periodically issued health-based air quality guidelines to assist governments and civil society reduce human exposure to air pollution and its adverse effects. The main aim is to offer quantitative health-based recommendations for air quality management, expressed as long- or short-term concentrations for several key air pollutants. Exceeding the air quality guideline (AQG) levels is associated with important risks to public health. These guidelines are not legally binding standards and have no obligatory character. However, they provide WHO Member States with an evidence-informed tool that they can implement into national programs to reduce levels of air pollutants to decrease the enormous health burden from exposure to air pollution worldwide.

Table D.1: Recommended air quality guidelines for each pollutant [5]

Pollutant	Guideline value	Average time	Guideline reference
$PM_{2.5}$	5 μg/m³	Annual	WHO 2021
	15 μg/m³	24-hour	
PM_{10}	15 μg/m³	Annual	WHO 2021
	45 μg/m³	24-hour	
Carbon monoxide (CO)	4 μg/m³	24-hour	WHO 2021
Nitrogen dioxide (NO_2)	10 μg/m³	Annual	WHO 2021
	25 μg/m³	24-hour	
Sulfur dioxide (SO_2)	40 μg/m³	24-hour	WHO 2021
Formaldehyde	0.1 μg/m³	30-minute	WHO 2010
Polycyclic aromatic hydrocarbons	8.7×10^{-5} per ng/m³		WHO 2010
Radon	100 Bq/m³		WHO 2010
Lead	0.5 μg/m³	Annual	WHO Regional Office for Europe, 2000

D.3 Studies of the effect of ambient air pollution on health

In the last century, increased combustion of fossil fuels and the continuous intensification of traffic are responsible for the progressive change in the atmospheric composition, which negatively affects the quality of life.

One of the main aspects is the influence of vehicle exhaust gases on health, to which children are particularly sensitive. Exhaust gases contain over 200 species of pollutants, some of which are: CO_2, NO_x, CO, SO_x, low molecular weight hydrocarbons, aldehydes (formaldehyde, acetaldehyde, acrolein), benzene, 1,3 butadiene, polycyclic hydrocarbons, oxidative component particles (elemental carbon, adsorbed aromatic hydrocarbons, small amounts of sulfate, nitrate, metals, and other elements), etc.

Although the reduced economic activity during the recession led to a reduction of atmospheric emissions, it is generally considered that automobile transport in Europe is responsible for harmful levels of air pollutants and a quarter of greenhouse gas emissions in the European Union. Standards "Euro" for vehicles achieved some success but did not significantly reduce NO_2.

Air pollutants, such as carbon monoxide (CO), sulfur dioxide (SO_2), nitrogen oxides (NOx), volatile organic compounds (VOCs), ozone (O_3), heavy metals, and particulate matter ($PM_{2.5}$ and PM_{10}), differ in their chemical composition, reaction properties, time of disintegration and ability to diffuse in long or short distances. Outdoor air pollution is a major environmental health problem affecting everyone in low-, middle-, and high-income countries because it can cause respiratory and other diseases and are important sources of morbidity and mortality [9]. These effects of air pollutants on human health and their mechanism of action will be briefly discussed in the following.

Air pollution has acute and chronic effects on human health, affecting several different systems and organs. It ranges from minor upper respiratory irritation to chronic respiratory and heart disease, lung cancer, acute respiratory infections in children and chronic bronchitis in adults, aggravating pre-existing heart and lung disease, or asthmatic attacks. In addition, short- and long-term exposures have also been linked with premature mortality and reduced life expectancy.

WHO estimates that in 2019, some 37% of outdoor air pollution-related premature deaths were due to ischaemic heart disease and stroke, 18% and 23% of deaths were due to chronic obstructive pulmonary disease and acute lower respiratory infections, respectively, and 11% of deaths were due to cancer within the respiratory tract. Ambient (outdoor) air pollution in both cities and rural areas was estimated to cause 4.2 million premature deaths worldwide per year in 2019; this mortality is due to exposure to fine particulate matter, which causes cardiovascular and respiratory disease and cancers.

Across the EU, it is common to have air pollution levels that are higher than the latest WHO recommendations. Still, there are signs of improvement, but some facts are pointed out below:

- In 2021, 97% of the urban population was exposed to concentrations of fine particulate matter above the health-based guideline level set by the World Health Organization.
- Every year, over 1,200 deaths in people under 18 years of age are estimated to be caused by air pollution in EEA member and collaborating countries [10].
- Data from 2021 show that Central-eastern Europe and Italy reported the highest concentrations of particulate matter, primarily due to the burning of solid fuels for domestic heating and their use in industry.
- All EU countries reported ozone and nitrogen dioxide levels above the health-based guideline levels set by the World Health Organization.
- Around 275,000 premature deaths are caused by fine particulate matter and 64,000 by nitrogen dioxide (NO2) each year.
- Overall, 97% of the EU's urban population was exposed to levels of fine particulate matter above the latest guidelines set by WHO in 2021.

The adverse effects of exposure to air pollution are a global public health concern in both developing and developed nations, as children and young people are particularly vulnerable to the effects of air pollution.

Epidemiological studies are the most indicative of evaluating the health effects of air pollution. One of the most vulnerable, appropriate for study contingent, are children in preschool and early school age because they spend more time outdoors, have a higher intensity of metabolic processes, and aspirate a relatively larger volume of air than adults. At the same time, they have not acquired bad habits (smoking, alcohol consumption, etc.) yet and are not exposed to industrial hazards. In a large complex of negative health effects of exhaust emissions, most clearly stand out violations in respiratory function, cardiovascular and immune system, hematopoietic, and others.

A large study involving children of preschool and early school age in 6 cities of northern China showed a strong positive correlation between respiratory symptoms (cough, difficulty breathing, wheezing, and phlegm) and total-suspended dust, sulfur dioxide, and nitrogen levels.

Particularly emphasized is the relationship between nitrogen dioxide and ozone and provoking or exacerbating respiratory diseases with an obstructive syndrome, asthma first. A prime example is cases where even temporarily reduced intensity to vehicle traffic reduces respiratory symptoms of the upper respiratory tract.

Air pollution impacts physical and mental development in childhood and exacerbates respiratory conditions like asthma and Seasonal Allergic Rhinitis (SAR), more commonly referred to as hay fever. Hay fever is the most common chronic condition in children and is most prevalent among school students. There is increasing evidence that air pollutants such as ozone (O3) may enhance the allergenicity of pollen, which in turn may impact cognitive development.

D.4 Health and environmental impact of air pollution

Air quality is a major concern for Europeans and is an area where the EU has been particularly active for more than 30 years. The EU's key objective regarding air quality is "to achieve levels of air quality that do not result in unacceptable impacts on, and risks to, human health and the environment." The questions in the annual Flash Eurobarometer survey are designed to support this work by providing greater insight into the European public's views on air quality and air pollution.

The Eurobarometer survey is designed to examine:

- the level of knowledge about air quality problems;
- the perceived seriousness of air quality problems and perceived changes in the quality of air over the past ten years;
- the perceived impact of various sectors and activities on air quality;
- the main threats to air quality;
- environmentally friendly energy and transport options;
- individual and other actions to reduce air quality problems;
- and many others.

The survey results in 2022 reveal that air quality is still a serious concern for European citizens. All raw data from the survey are freely available and can be accessed online.

- While most Europeans do not feel well-informed (60%), nearly half of the respondents believe that air quality has deteriorated in the last ten years (47%).
- Most Europeans think that health conditions such as respiratory diseases (89%), asthma (88%), and cardiovascular diseases are serious problems in their countries resulting from air pollution. The Eurobarometer reveals that citizens lack information about air quality problems in their country.
- Most Europeans remain poorly informed about the existing EU air quality standards, as only a minority of respondents (27%) have heard of them.
- Nevertheless, a large majority of the respondents (67%) aware of EU air quality standards say that they should be strengthened.

A screening questionnaire for the assessment of the air pollution perception and the risk of exposure to outdoor and indoor air pollution

For the questionnaire development, a pool of items based on many standardized survey recommendations was used in addition to mechanisms from similar studies on air pollution protection. Items were carefully written to minimize ambiguity and increase comprehension. In total, the item pool was comprised of 25 items. The questionnaire is a promising instrument to assess the attitudes and perceptions of the population to air pollution and the risk of exposure to outdoor and indoor pollution. This questionnaire could be used by scientists, researchers, authorities, and health promotion planners to develop and implement air pollution protection promotion programs.

Questionnaire A – main items

Please, read all questions and answer by ticking the box or by providing a brief explanation where appropriate.

The survey is anonymous, and we assure you that the confidentiality of your individual responses will be kept.

1. Gender

☐ male ☐ female

2. Age

☐ under 3 years old ☐ 3-7 years old ☐ 8-14 years old

☐ 15-20 years old ☐ 21-30 years old ☐ 31-40 years old

☐ 41-50 years old ☐ 51-60 years old ☐ over 60 years old

3. What region do you live in?

Country..... Settlement .....

4. Would you say you live in …"

☐ rural area ☐ village ☐ small town

☐ medium-sized town ☐ large town/city

5. Employment

☐ pupil ☐ student ☐ self-employed ☐ employee ☐ manual worker ☐ without a professional activity ☐ refusal ☐ other

Please, specify.

6. How many people aged 15 years or more live in your household, yourself included?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ other .... *Please, specify.*

7. What is the average monthly income of your family (per member)?

☐ up to 300 euros ☐ 300-600 euros ☐ 600-1000 euros

☐ 1000-1500 euros ☐ more than 1500 euros ☐ other ..... *Please, specify.*

8. What type of domestic heating do you have?

☐ electric heater ☐ gas heater ☐ air conditioner ☐ furnace

☐ wood/pellet heating ☐ solar heating ☐ other ..... *Please, specify.*

9. How informed do you feel about air quality problems in your country?

☐ very well informed ☐ well informed ☐ not well informed

☐ not informed at all ☐ other ..... *Please, specify.*

10. Do you think that, over the last 10 years, the air quality in your country has …?

☐ improved ☐ stayed the same ☐ deteriorated

☐ other ..... *Please, specify.*

11. How much impact do you think each of the following has on air quality in your country? Does it have a large impact, a moderate impact, a little impact, or no impact at all?

	A large impact	A moderate impact	A little impact	No impact at all
Residential energy use (e.g., coal and wood for heating of individual households)				
Agriculture- emissions from farms, fertilizers and burning of agricultural waste				
Emissions from cars and trucks				
Emission from international transport (e.g. ships and airplanes)				
Emissions from industrial production (steel, cement, pulp, paper, etc.) and from fossil fuel power stations				
Landscape				
Rivers / lakes				
Clean air				
Other				

12. Which three of the following emitters do you believe are the main threats to air quality in your country?

☐ cross-border emissions from other countries/regions
☐ transport activities
☐ electricity and heat production
☐ natural pollutants (sea salt, desert sand, volcanic ash)
☐ industrial activities
☐ emissions from individual households
☐ emissions from farms
☐ other ..... *Please, specify.*

13. Which two of the following car fuel systems do you consider the most environmentally friendly from an air quality perspective?

☐ gasoline ☐ diesel ☐ biofuel
☐ hybrid electric/gasoline cars ☐ hybrid electric/diesel cars
☐ electric cars ☐ other ..... *Please, specify.*

14. Which two of the following energy systems for domestic heating do you consider the most environmentally friendly from an air quality perspective?

☐ oil ☐ gas ☐ coal ☐ biomass (wood)
☐ biomass (pellets) ☐ electricity ☐ district heating
☐ other ..... *Please, specify.*

15. There are different ways to reduce harmful emissions to the air. To reduce these problems, have you done any of the following in the last two years? Please select all applicable.

☐ You changed your housing heating system from higher-emitting (e.g., coal, oil, or wood-fired) to lower-emitting (e.g., natural gas, pellets, electricity)

☐ You replaced older energy-using equipment (hot water boiler, oven, dishwasher, etc.) with newer ones having better energy efficiency rating (e.g., A+++ for energy efficiency)

☐ You frequently used public transport, cycling or walking instead of your car

☐ You bought a low-emission car

☐ You bought low-emitting products to fuel your open fire or barbeque (i.e., briquettes instead of coal)

☐ other ..... *Please, specify.*

16. Would you say that the following is a very serious problem, a fairly serious problem, not a very serious problem, or not a serious one in your country?

	A very serious problem	A fairly serious problem	Not a very serious problem	Not a serious problem at all
Respiratory diseases (e.g., lung diseases)				
Cardio-vascular diseases (heart diseases)				
Asthma and allergy				
Acidification (acid rains, affecting forests, etc.)				
Eutrophication (an increase of organic matter in an ecosystem, such as excessive growth of algae causing fish die-offs in rivers or lakes)				

17. In your opinion, is each of the following doing too much, doing about the right amount, or not doing enough to promote good air quality in your country?

	Doing too much	Doing the right amount	Not doing enough	I don't know
Households				
Farmers				
Energy producers				
Car manufacturers				
Public authorities				

18. In your opinion, how can air pollution challenges be best addressed?

☐ at local level ☐ at national level ☐ at European level

☐ other .....*Please, specify.*

19. How would you rate the overall air quality in your city/town/village now compared to last year?

☐ much better ☐ a little better ☐ about the same ☐ a little worse

☐ much worse ☐ other .....

20. What are the main causes of air pollution in your city? Please select all applicable.

☐ construction ☐ industrial sources/manufacturing facilities ☐ motor vehicles
☐ household cooking and heating ☐ increasing use of air conditioners ☐ population growth
☐ power plants ☐ smoke of cigarettes ☐ waste disposal ☐ burning of waste
☐ pollution from other regions ☐ other .....

21. To what extent is the air pollution affecting you?

☐ Breathlessness/having more difficulty in breathing
☐ Doing less outdoor activities
☐ Doing more to look after my skin
☐ Doing more to stay healthy
☐ Feeling depressed
☐ Irritation to eyes/nose/throat
☐ Skin problems
☐ Wanting to move to other less polluted places
☐ Asthma incidence
☐ Poor visibility
☐ Worrying about the living environment
☐ other ..... *Please, specify.*

22. Your home is located …?

☐ in a quiet area, low car traffic
☐ in a noisy area, heavy car traffic
☐ in a noisy area, due to different from traffic source
☐ other ..... *Please, specify.*

23. Can you smell any exhausting gases from car traffic in your home?

☐ yes, everyday ☐ yes, often ☐ rare ☐ other ..... *Please, specify.*

24. To what extent do you experience pollution from vehicles (noise, exhaust gases, etc.) in your home??

☐ very high ☐ medium ☐ low ☐ other ..... *Please, specify.*

25. Do your family have problems when sleeping at night (waking by the traffic noise)?

☐ yes, very often ☐ often ☐ rare ☐ other ..... *Please, specify.*

Part B – a special section about children's health

Please, read all questions and answer by ticking the box or by providing a brief explanation where appropriate.

The survey is anonymous, and we assure you that the confidentiality of your individual responses will be kept.

1. Gender

☐ male ☐ female

2. Age

☐ below 3 years old ☐ 3-7 years old ☐ 8-15 years old ☐ above 15 years

3. Children weight

☐ at birth …… …….. ☐ at present …….

4. Mother's age at the birth of the child

☐ up to 20 years old ☐ 21-30 years old ☐ 31-40 years old

☐ 41-50 years old ☐ above 50 years old

5. How long has the baby been breastfed (in months)?

☐ up to 1 month ☐ 1-3 months ☐ 3-6 months

☐ 6-9 months ☐ 9-12 months ☐ other ..... *Please, specify.*

6. Are there cigarette smokers in thy family? How many?

☐ 1 ☐ 2 ☐ 3 ☐ other ..... *Please, specify if the mother is.*

7. Are there pets in your home? How many?

☐ yes ☐ no ☐ other ..... *Please, specify.*

8. Do any parents or brothers/sisters have an allergic disease?

☐ yes ☐ no ☐ other ..... *Please, specify.*

9. Does your child (respondent) has an allergic disease?

☐ yes ☐ no ☐ other ..... *Please, specify.*

10. Has your child (respondent) passed any serious disease until now (need of hospitalization)?

☐ yes ☐ no other ..... *Please, specify.*

11. Does your child suffer from respiratory diseases (runny noses, bronchitis, pneumonia) more often than four times a year

☐ yes ☐ no ☐ other ..... *Please, specify.*

12. Have you registered some of the following symptoms in your child (respondent) during the last six months?

	yes	no	I don't know
persistent cough			
whistling wheezing			
dry cough at night			
hay fever			
attacks of difficulty in breathing (asthma)			
flu or other disease affecting the respiratory system			
bloodshot eyes (conjunctivitis)			

E CURRICULUM

This part of the handbook represents a curriculum for the "Advanced Big Data Processing Technologies" course. This curriculum is provided by the Faculty of Natural Sciences team at Matej Bel University in Banská Bystrica, Slovakia. This partner has implemented this course, and all partners will implement it during project sustainability.

University: Matej Bel University, Banská Bystrica, Slovakia
Faculty: Faculty of Natural Sciences
Code: DEK FPV/2d-fpv-401
Course title: Advanced Big Data Processing Technologies in Natural Sciences
Type, workload and methods of educational activities: Course type: optional Recommended workload: 2 hours of seminars/week Method of study: combined Form of study: full time Number of credits: 3 Recommended semester: 2nd semester of Master studies
Degree of study: second (Master)
Prerequisity courses: no prerequisities
Conditions for passing and completing the course: a) continuous assessment: active participation in exercises, completion of assigned tasks 100 % b) final assessment: 0 % The evaluation of the subject is in accordance with the classification scale determined by the UMB study regulations.
Learning outcomes: 1. Students will gain knowledge and skill in the areas of: 2. Introduction to data processing and analysis 3. Introduction to basic data analysis tasks – regression and classification 4. Introduction to work with Big Data – data sampling methods 5. Statistical methods in analysis of data 6. Basics of Exploratory Data Analysis – theory and practice 7. Introduction to fuzzy sets 8. Fuzzy sets and regression task 9. Fuzzy sets and classification tasks 10. Introduction to Neural Networks During the course student will gain experience in work with: 1. MATLAB software tool 2. R software tool

X REFERENCES

References for sections 1 – 4:

- C.J. Date. An Introduction to Database Systems (8th. ed.). Addison-Wesley Longman Publishing Co., 2003. ISBN: 978-0-321-19784-9
- Felix Kutsanedzie, Sylvester Achio, Edmund Ameko. Practical Approaches to Measurements, Sampling Techniques and Data Analysis. Science Publishing Group, 2016. ISBN: 978-1-940366-58-6.
- William J. Lammers, Pietro Badia. Fundamentals of Behavioral Research Textbook. Online: https://uca.edu/psychology/fundamentals-of-behavioral-research-textbook/
- Jimin Quian et al. Introducing self-organized maps (SOM) as a visualization tool for materials research and education. Results in Materials, Volume 4, 2019, ISSN 2590-048X.
- Naseer Raheem. Big Data: A tutorial-based approach. Chapman and Hal-l/CRC, 2019. ISBN: 978-0-367-67024-5
- Lior Rokach, Oded Maimon. Data mining with decision trees. 2015.
- Steven S. Skiena. The Data Science Design Manual. Springer, 2017. ISBN: 978-3-319-55443-3
- Karthik Ramasubramanian, Abhishek Singh. Machine Learning Using R. Springer, 2019. ISBN: 978-1-4842-4214-8
- Patrik Očenáš. Parallel and distributed methods of big data sampling (in Slovak). 2023.
- Bianka Modrovičová. Decision trees for sizable graph datasets (in Slovak). 2023.
- Aneta Szoliková. Explorative data analysis in document databases (in Slovak). 2023.
- Adam Dudáš, Bianka Modrovičová. Decision Trees in Proper Edge k-coloring of Cubic Graphs. In Proceedings of 33rd FRUCT conference. 2023.

References for sections 5 – 8:

- ZADEH, L. A. Fuzzy Sets. In: Information and Control, 8, 1965, 338-353.
- MICHALÍKOVÁ, A.: Fuzzy množiny v informatike. rec. Mirko Navara, Martin Kalina, Martin Klimo. Belianum. Matej Bel University in Banská Bystrica, 1, 2020, 206p. ISBN 978-80-557-1707-4
- Sendai Subway. Japan Visitor [cit. 2023-02-02]. Online: https://www.japanvisitor.com/japan-transport/sendai-subway
- RUAN D.: Fuzzy Logic Applications in Nuclear Industry. Fuzzy Logic Foundations and Industrial Applications. 1996, 8, ISBN 978-1-4612-8627-1.
- TAKAGI, T., SUGENO, M. Fuzzy Identifications of Fuzzy Systems and its Applications to Modelling and Control. In: IEEE Transactions on Systems, Man, and Cybernetics, 15(1), 1985, 116-132.
- ROSS, T. J. Fuzzy Logic with Engineering Applications. John Wiley & Sons, 2005, 585s., ISBN 9780470743768.
- ZADEH, L. A., The Concept of a Linguistic Variable and its Application to Approximate Reasoning - 1, In: Information Sciences, 8, 1975, 199–249.

References for sections 9

- Ahmed, Z. H. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. International Journal of Biometrics & Bioinformatics (IJBB), 3(6), 96.
- Aktaş, M., Yetgin, Z., Kılıç, F., & Sünbül, Ö. (2022). Automated test design using swarm and evolutionary intelligence algorithms. Expert Systems, 39(4), e12918.
- Bartz-Beielstein, T., Branke, J., Mehnen, J., & Mersmann, O. (2014). Evolutionary algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(3), 178-195.
- Bertsimas, D., & Tsitsiklis, J. (1993). Simulated annealing. Statistical science, 8(1), 10-15.
- Blickle, T. (2000). Tournament selection. Evolutionary computation, 1, 181-186.
- Cui, Y., Geng, Z., Zhu, Q., & Han, Y. (2017). Multi-objective optimization methods and application in energy saving. Energy, 125, 681-704.
- De La Iglesia, B. (2013). Evolutionary computation for feature selection in classification problems. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 3(6), 381-407.
- Gaivoronski, A. A., Lisser, A., Lopez, R., & Xu, H. (2011). Knapsack problem with probability constraints. Journal of Global Optimization, 49, 397-413.
- Glover, F., & Laguna, M. (1998). Tabu search (pp. 2093-2229). Springer US.
- Hansen P, Mladenovi'c N (1999) An introduction to variable neighborhood search. In: Voß S, Martello S, Osman IH, Roucairol C (eds) Metaheuristics: advances and trends in local search paradigms for optimization, chapter 30. Kluwer Academic Publishers, Dordrecht, pp 433–458
- Hayyolalam, V., & Kazem, A. A. P. (2020). Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 87, 103249.

- Hinson, J. M., & Staddon, J. E. R. (1983). Matching, maximizing, and hill-climbing. Journal of the experimental analysis of behavior, 40(3), 321-331.
- Holland JH. Outline for a logical theory of adaptive systems. J ACM. 1962;9(3):297–314
- Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. SIAM journal on computing, 2(2), 88-105.
- Hoos, H. H., & Stützle, T. (2004). Stochastic local search: Foundations and applications. Elsevier.
- I. Rechenberg, Cybernetic solution path of an experimental problem. Royal Air-craft Establishment, Library Translation 1122, Farnborough, Reprint in: D.B. Fogel (Ed.), Evolutionary Computation, The Fossil Record, IEEE Press, Piscataway, NJ, 1965, pp. 301–309
- I. Rechenberg, Evolutionsstrategie—Optimisierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart, 1973
- Kılıç, F., Yılmaz, İ. H., & Kaya, Ö. (2021). Adaptive co-optimization of artificial neural networks using evolutionary algorithm for global radiation forecasting. Renewable Energy, 171, 176-190.
- Kiliç, F., & Gök, M. (2013). A public transit network route generation algorithm. IFAC Proceedings Volumes, 46(25), 162-166.
- Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K., & China, H. (2013). Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. gene, 7(33), 8.
- Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. Knowledge-based systems, 96, 120-133.
- Rossi, F., Van Beek, P., & Walsh, T. (Eds.). (2006). Handbook of constraint programming. Elsevier.
- Salkin, H. M., & De Kluyver, C. A. (1975). The knapsack problem: a survey. Naval Research Logistics Quarterly, 22(1), 127-144.
- Sharifi, A. A., & Aghdam, M. H. (2019). A novel hybrid genetic algorithm to reduce the peak-to-average power ratio of OFDM signals. Computers & Electrical Engineering, 80, 106498.
- Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., & Zhao, W. (2022). Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 114, 105082.
- Yang, J., & Soh, C. K. (1997). Structural optimization by genetic algorithms with tournament selection. Journal of computing in civil engineering, 11(3), 195-200.

References for section 10:

- Basic Neural Networks 1 - https://docs.google.com/a/atu.edu.tr/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxpaHNhbnlhc3NpbjJ8Z3g6NGY4MjNjN2Y4ZTdhNWM2MQ
- Basic Neural Networks 2 - http://www.cs.stir.ac.uk/courses/ITNP4B/lectures/
- Basic Neural Networks 3

  https://www.cs.bham.ac.uk/~jxb/inn.html
- Basic Neural Network 4

  https://www.fer.unizg.hr/en/course/neunet_a/lecture_notes
- Basic Neural Network 5

  http://users.monash.edu/~cema/courses/FIT3094/lecturePDFs/

References for section 11:

- Paluszek, M., Thomas, S. Matlab machine learning recepies. 2019. Plainsboro, NJ, USA. ISBN-13 (pbk): 978-1-4842-3915-5. DOI 10.1007/978-1-4842-3916-2.
- Kim, P. MATLAB Deep Learning. With Machine Learning, Neural Networks and Artificial Intelligence. 2017. Apress Korea ISBN-13 (pbk): 978-1-4842-2844-9. DOI 10.1007/978-1-4842-2845-6.
- Get Started with Matlab. https://www.mathworks.com/help/matlab/getting-started-with-matlab.html
- Iris Clustering. https://www.mathworks.com/help/deeplearning/ug/iris-clustering.html

References for Appendices:

- Fisher, R.A. (1936) "The use of multiple measurements in taxonomic problems". Annual Eugenics, 7, Part II, pages 179-188
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, pages 431-433
- Duda, R.O., Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1, page 218
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recogni-

tion in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, pages 67-71
- https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-3/data-products
- https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-4/data-products
- https://climexp.knmi.nl/
- https://www.uradmonitor.com/
- Velea L, Udriștioiu MT, Puiu S, Motișan R, Amarie (2023)D. A Community-Based Sensor Network for Monitoring the Air Quality in Urban Romania. Atmosphere; 14(5):840. https://doi.org/10.3390/atmos14050840
- https://bookdown.org/floriandierickx/bookdown-demo/climate-data-from-models.html#differences-between-climate-projections-predictions-and-scenarios
- https://ec.europa.eu/eurostat/web/climate-change/database
- https://ourworldindata.org/
- https://ourworldindata.org/data-review-air-pollution-deaths
- •https://ourworldindata.org/outdoor-air-pollution#outdoor-air-pollution-deaths-by-age
- •https://www.who.int/health-topics/air-pollution#tab=tab_1
- •https://www.eea.europa.eu/en/topics/in-depth/air-pollution
- •https://www.who.int/teams/environment-climate-change-and-health/air-quality-and-health/health-impacts/types-of-pollutants
- https://www.who.int/publications/i/item/9789240034228
- •https://apps.who.int/iris/bitstream/handle/10665/345329/9789240034228-eng.pdf
- EEA, 2012, The contribution of transport to air quality, EEA Report no. 10/2012, European Environment Agency.
- EEA. A closer look at urban transport TERM 2013: transport indicators tracking progress towards environmental targets in Europe EEA Report No 11/2013 Copenhagen, ISSN 1725-9177.
- http://dx.doi.org/10.1016/j.envpol.2007.06.012
- https://www.who.int/health-topics/air-pollution#tab=tab_1
- Report no. 05/2022, Air quality in Europe 2022. doi: 10.2800/488115. https://www.eea.europa.eu/publications/air-quality-in-europe-2022
- Xin Zhang, X. Chen, Xiaobo Zhang. The impact of exposure to air pollution on cognitive performance. Proc. Natl. Acad. Sci. Unit. States Am., 115 (2018), pp. 9193-9197, 10.1073/pnas.1809474115
- J. Currie, J.S.G. Zivin, J. Mullins, M.J. Neidell. What do we know about short and long term effects of early life exposure to pollution? NBER Work. Pap., 6 (2013), pp. 217-247, 10.3386/w19571
- Escamilla-Nuñez M-C., Barraza-Villarreal A., Hernandez-Cadena L., Moreno-Macias H., Ramirez-Aguilar M., Sienra-Monge J-J., Cortez-Lugo M., Texcalac J-L., del Rio-Navarro B., Romieu I. Traffic-Related Air Pollution and Respiratory Symptoms Among Asthmatic Children, Resident in Mexico City: The EVA Cohort Study. http://www.medscape.com/viewarticle/585875.
- Juvin P., Fournier T., Boland S. et al. Diesel particles are taken up by alveolar type II tumor cells and alter cytokines secretion. Arch Environ Health. 2002; 57(1):53-60.
- Le Tertre A., S. Medina, E. Samoli et al: Short term effects of particulate air pollution on cardiovascular disease in eight European cities. J. Epidemiol Community Health, 2002; 56, (10):773-9.
- Nordling E., Berglind N., Melén E., Emenius G., Hallberg J., Nyberg F., Pershagen G., Svartengren M., Wickman M., Bellander T. Traffic related air pollution and childhood respiratory symptoms, function and allergies. Epidemiology. 2008; 19(3):401-8.
- Pan G., Zhang S., Feng Y., Takahashi K., Kagawa J., Yu L., Wang P., Liu M., Liu Q., Hou S., Pan B., Li J. Air pollution and children's respiratory symptoms in six cities of Northern China. Respiratory Medicine 2010;104(12):1903–11.
- Richardson E.A., Pearce J., Tunstall H., Mitchell R., Shortt N.K.: Particulate air pollution and health inequalities: a Europe-wide ecological analysis. Int J Health Geogr 2013;12:34
- I. Jáuregui, J. Mullol, I. Dávila, M. Ferrer, J. Bartra, A. Del Cuvillo, J. Montoro, J. Sastre, A. Valero. Allergic rhinitis and school performance. J Investig. Allergol. Clin. Immunol., 19 (2009), pp. 32-39
- D.P. Skoner. Allergic rhinitis: definition, epidemiology, pathophysiology, detection, and diagnosis. J. Allergy Clin. Immunol., 108 (2001), pp. 2-8, 10.1067/mai.2001.115569
- I. Beck, S. Jochner, S. Gilles, M. McIntyre, J.T.M. Buters, C. Schmidt-Weber, H. Behrendt, J. Ring, A. Menzel, C. Traidl-Hoffmann. High environmental ozone levels lead to enhanced allergenicity of birch pollen. PloS One, 8 (2013), 10.1371/journal.pone.0080147
- P. Sturdy, S. Bremner, G. Harper, L. Mayhew, S. Eldridge, J. Eversley, A. Sheikh, S. Hunter, K. Boomla, G. Feder, K. Prescott, C. Griffiths. Impact of asthma on educational attainment in a socioeconomically deprived population: a study linking health, education and social care datasets. PloS One, 7 (2012), pp. 1-8, 10.1371/journal.pone.0043977
- https://europa.eu/eurobarometer/surveys/detail/2660
- https://data.europa.eu/data/datasets/s2660_97_2_sp524_eng?locale=en
- https://www.surveymonkey.com/r/airpollutionperceptionsurvey
- https://apps.who.int/iris/rest/bitstreams/1350812/retrieve
- https://www.ab.gov.tr/files/ardb/evt/Attitudes_of_Europeans_towards_air_quality_2013.pdf