# An Introduction to Business Analytics

Copyright © 2019 Ger Koole
All rights reserved
MG books, Amsterdam
ISBN 978 90 820179 3 9
Cover design: Ingrid Brandenburg & Luciano Picozzi

# An Introduction to Business Analytics

Ger Koole

MG books
Amsterdam

# Preface

Books on Business Analytics (BA) typically fall into two categories: managerial books without any technical details, and very technical books, written for BA majors who already have a background in advanced mathematics or computer science. This book tries to fill the gap by discussing BA techniques at a level appropriate for readers with a less technical background. This makes it suitable for many different audiences, especially managers who want to better understand the work of their data scientists, or people who want to learn the basics of BA and do their first BA projects themselves.

The full range of BA-related topics is covered: from the many different techniques to an overview of managerial aspects; from comparisons of the usefulness of different techniques in different situations to their historical context. While working with this book, you will also learn appropriate tooling, especially R and a bit of Excel. There are exercises to sharpen your skills and test your understanding.

Because this book contains a large variety of topics, I sought advice from many experts. I am especially indebted to Sandjai Bhulai, Bram Gorissen, Jeroen van Kasteren, Diederik Roijers and Qingchen Wang for their feedback on scientific issues and Peggy Curley for editing.

Business Analytics is a young field in full development, which uses aspects from various fields of science. Although I tried to integrate the knowledge from many fields, it is unavoidable that the content will be biased based on my background and experience. Please do not hesitate to send me an email if you have any ideas or comments to share. I sincerely hope that reading this book is a rewarding experience.

All chapters can be read independently, but I advise to read Chapter 1 first to understand the connections between the chapters. The index at the end can be helpful for unknown terms and abbreviations.

Ger Koole Amsterdam/Peymeinade, 2016–2019

# Contents

# Chapter 1

# Introduction

This chapter explains business analytics and data science without going into any technical detail. We will clarify the meaning of different terms used, put the current developments in a historical perspective, give the reader an idea of the potential of business analytics (BA), and give a high-level overview of the steps and pitfalls in implementing a BA strategy.

**Learning outcomes** On completion of this chapter, you will be able to:

- describe in non-technical terms the field of business analytics, the different steps involved, the connections to other fields of study and its historical context
- reflect on the skills and knowledge required to successfully apply business analytics in practice

## 1.1 What is business analytics?

According to Wikipedia, "Business analytics refers to the skills, technologies, practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning." In short, BA is a rational, fact-based approach to decision making. These facts come from data, therefore BA is about the science and the skills to turn data into decisions. The science is mostly *statistics*, *artificial intelligence* (*data mining* and *machine learning*), and *optimization*; the skills are computer skills, communication skills, project and change management, etc.

It should be clear that BA by itself is not a science. It is the total set of knowledge that is required to solve business problems in a rational way. To be a successful business analyst, experience in BA projects and knowledge of the business areas that the data comes from (such as healthcare, advertising, finance) is also very valuable.

BA is often subdivided into three consecutive activities: *descriptive analytics*, *predictive analytics*, and *prescriptive analytics*. During the descriptive phase, data is analyzed and patterns are found. The insights are consequently used in the predictive phase to predict what is likely to happen in the future, if the situation remains the same. Finally, in the prescriptive phase, alternative decisions are determined that change the situation and which will lead to desirable outcomes.

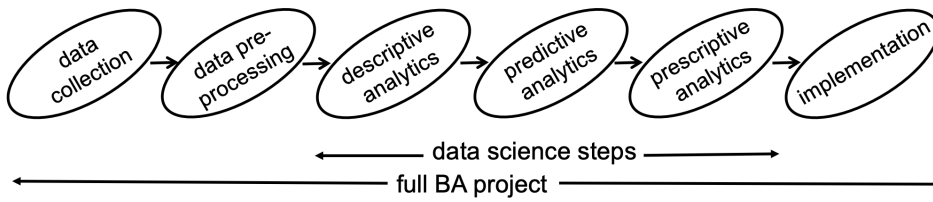
**Example 1.1** *A hotel chain analyzes its reservations to look for patterns: which are the busiest days of the week? What is the impact of events in the city? Is there a seasonal pattern? Etc. The outcomes are used to make a prediction for the revenue in the upcoming months. By changing the pricing of the rooms in certain situations (such as sports events), the expected revenue can be maximized.*

Analytics can only start when there is data. Certain organizations already have a centralized *data warehouse* in which relevant current and historical data is stored for the purpose of reporting and analytics. Setting up such a data warehouse and maintaining it is part of the *business intelligence* (BI) strategy of a company. However, not all companies have such a centralized database, and even when it exists it rarely contains all the information required for a certain analysis. Therefore, data often needs to be collected, cleansed and combined with other sources. Data collection, cleansing and further pre-processing is usually a very time-consuming task, often taking more time than the actual analysis.

**Example 1.2** *In the hotel* revenue management *example above we need historical data on reservations but also data on historical and future events in the surroundings of the hotel. There are many reasons why this data can be hard to get: reservation data may only be stored at an aggregated level, there may have been changes in IT systems which overrode previously collected data, there may be no centrally available list with events, etc. Many organizations assume they already have all the data required, but as soon as the data scientist asks for reservation data combined with the date the booking was made or the event list from the surrounding area, the hotel might find out that they lack data.*

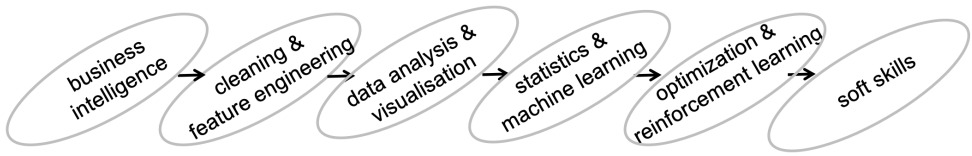
Therefore, data collection and pre-processing are always the first steps of a BA project. Following the data collection and pre-processing the real data science steps begin with descriptive analytics. Moreover, a BA project does not end with prescriptive analytics, i.e., with generating an (optimal) decision. The decision has to be implemented, which requires various skills, such as knowledge of change management.

To summarize, we distinguish the following steps in a BA project:




The model above suggests a linear process, but in practice this is rarely the case. At many of the steps, depending on the outcome, you might revisit earlier steps. For example, if the predictions are not accurate enough for a particular aaplication then you might collect extra data to improve them. Furthermore, not all BA projects include prescriptive analytics, many projects have insight or prediction as goal and therefore finish after the descriptive or predictive steps.

The major scientific fields of study corresponding to these BA steps are:




Next to cleansing, *feature engineering* is an important part of data preparation, to be discussed later. During descriptive analytics you get an understanding of the data. You visualize the data and you summarize it using the tool of statistical data analysis. Getting a good understanding is crucial for making the right choices in the consecutive steps.

Following the descriptive analytics a BA project continues with predictive analytics. A target value is specified which we want to predict. Based on the data available, the parameters of the selected predictive method are determined. We say that the model is *trained* on the data. The methods originate from inferential statistics and machine learning, which have their respective roots in mathematics and computer science. Although the approach and the background of these fields are quite different, the techniques

largely overlap.

**Example 1.3** *A debt collection agency wants to use its resources, mainly calls to debtors, in a better way. It collects data on payments which is enriched by external data on household composition and neighborhood characteristics. After the data analysis and visualization a method is selected that predicts, given the characteristics of the dept and the actions taken by the agengy, the probability that the deptor will pay off their debt. In the prescriptive step, which is to be discussed next, the best action for each deptor is determined.*

Finally, during the prescriptive analytics phase, options are found to maximize a certain objective. Because the future is always unpredictable to a certain extent, optimization techniques often have to account for this randomness. The field that specializes in this is (mathematical) *optimization*. It overlaps partially with *reinforcement learning*, which has its roots in computer science. A special feature of reinforcement learning is that prediction and optimization are integrated: it combines in one method the predictive and prescriptive phases.

**Example 1.4** *Consider again Example 1.2 on hotel revenue management. After having studied the influence of events and for example intra-week fluctuations on hotel reservations in the descriptive step demand per price class is* forecasted *in the predictive step. These forecasts are input to an optimization algorithm that determines on a daily basis the prices that maximize total revenue.*

We end this section by discussing two terms that are closely related to BA: *Data science* and *big data*. Data science is an older term which has recently shifted in meaning and increased in popularity. It is a combination of different scientific fields all concerned with extracting knowledge from data, mainly data mining and statistics. Part of the popularity probably stems from the fact that the Harvard Business Review called a data scientist role "the sexiest job of 21st century", anticipating the huge demand for data scientists. The knowledge base of data scientists and business analysts largely overlap. However, the deliverable of BA is improved business performance, whereas data scientists focus more on methods and insights from data. Improved business performance requires optimization to generate decisions and *soft skills* to implement the decisions.

Finally, a few words on big data. Big data differentiates itself from regular data sets by the so-called 3 V's: *volume*, *variety*, and *velocity*. A data

**Box 1.1. From randomized trials to using already available data**
The traditional way to do scientific research in the medical and behavorial sciences is through (double-blind) *randomized trials*. This means that subjects (e.g., patients) have to be selected, and by a randomized procedure they are made part of the trial or part of the control group. It is called double blind when the subject and the researcher are both not aware of who is in which group. This kind of research set-up allows for a relatively simple statistical analysis, but it is often hard to implement and very time-consuming.
Nowadays, data can often be obtained from Electronic Health Records and other data sources. This eliminates the need for separate trials. However, there will be all kinds of statistical *biases* in the data, making it harder to make a fair comparison between treatments. For example, patients of a certain age or having certain symptoms might get more-often a certain treatment. This calls for advanced statistical methods to eliminate these biases. These methods are usually not taught in medical curricula, requiring the help of expert data scientists.

set is considered to be "big data" when the amount of data is too much to be stored in a regular database, when it lacks a homogeneous structure (i.e., free text instead of well-described fields), and/or when it is only available real-time. Big data requires adapted storage systems and analysis techniques in order to exploit it.

Big data now receives a lot of attention due to the speed at which data is collected these days. As more and more devices and sensors automatically generating data are connected to the internet (the *internet of things*) again, the amount of stored data doubles approximately every 3 years. However, most BA projects do not involve big data, but use with relatively small and structured data sets. It might have been the case that such a dataset had its origin in big data from which relevant information has been extracted.

**Example 1.5** *Cameras in metro stations are used to surveil passengers. Using image recognition software the numbers of passengers can be extracted, which can be used as input for a prediction method that forecasts future passenger volumes.*

## 1.2 Historical overview

Business analytics combines techniques from different fields all originating from their own academic background. We will touch upon the main constituent fields of statistics, artificial intelligence, *operations research*, and also BA and data science (DS).

Statistics is a mathematical discipline with a large body of knowledge developed in the pre-computer age. For many decades, statistics has been taught at universities without the use of any data sets. The central body of knowledge concerns the behavior of statistical quantities in limiting situations, for example when the number of observations approaches infinity. This is of a highly mathematical nature. More recently new branches of statistics have come into existence, many of which are more experimental in nature. However, quite often statistics is still taught as a mathematical discipline with a focus on the mathematics.

Artificial intelligence (AI) is a field within computer science that grew rapidly from the 1970s with the advent of computers.

Initial expectations were highly inflated. One believed, for example, that so-called *expert systems* would soon replace doctors in their work of diagnosing illnesses in patients. This did not happen and the attention for AI diminished. Today, the expectations are high again, largely due the fields of *data mining* and *machine learning* which are relevant for BA. They developed more recently when large data sets became available for analysis. Both fields of data mining and machine learning focus on learning from data and making predictions using what is learned. Machine learning focuses on predictive models, data mining more broadly on the process from data pre-processing to predictive analytics, with a focus on data-driven methods. The difference between statistics and machine learning are their origins and the more data-oriented approach of ML: Mathematicians want to prove theoretically that things work, computer scientists want to show it using data.

Operations research (OR) is about the application of mathematical optimization to decision problems in organizations. OR, sometimes called *management science*, and abbreviated as OR/MS, also raised big expectations, in the 1950s, following the first successes of the allied forces of OR being applied during World War II. The belief was that scientific methods would replace traditional management and turn it into a science. However, the impact at the strategic decision level remained very limited and OR applications are mainly found at the operational level. Quite often the application of OR would be to a logistical problem such as the routing of delivery vans, outside the scope of higher management. OR faces the same problems as statistics: it has been developed as a highly mathematical science, but it has a hard time adapting itself to the current situation in which data and tooling is easily available. Often it is still taught in a highly abstract mathematical way, limiting the potential impact in practice.

BA on the other hand, developed in organizations that realized that their data was not just valuable for their current operations, but also to gain insight and improve their processes. Starting in the 1990's, we saw more and more analysts working with data in organizations. An important difference with OR is that many executives do understand the value of analytics and adopt a company-wide BA strategy. A book by Davenport [8], who is an advocate of BA, also played a role in increasing the interest in the value of analytics to executives. Interestingly enough, the main example throughout the book is dynamic pricing in airlines, a typical OR success. The name OR is not mentioned once. This supports the opinion that some of these new areas are in fact rebranded old areas, it's old wine in a new bottle. Whether this is really true, or if there are fundamental differences between areas is not really relevant. The fact is that the availability of data, computers and software made the widespread use of BA possible. Finally, BA methods—also the ones originating from the mathematical sciences—are used on a huge scale in companies, institutions and research centers, offering countless opportunities for business analysts and data scientists.

DS as a term has been around for a long time. In the end of the last century it was mainly associated with statistics. Much like BA, the term became popular with the availability of large data sets. However, today it is moreoften associated with techniques from computer science such as machine learning. In contrast, BA is more often associated with mathematics and industrial engineering.

## 1.3 Non-technical overview

In this section we give a non-technical overview of the most often used techniques and explain some of the technical terms that are regularly used. This section by nature can only be an oversimplification of reality, but it will help to get a flavor of the totality of the field, which even professionals in the field sometimes do not have. The techniques we discuss in this section are summarized in Figure 1.1.

The four steps pre-processing, descriptive, predictive and prescriptive analytics, can also be described as follows:
- preparing the data set;
- understanding the data set;
- predicting a target value;
- maximizing the target value.

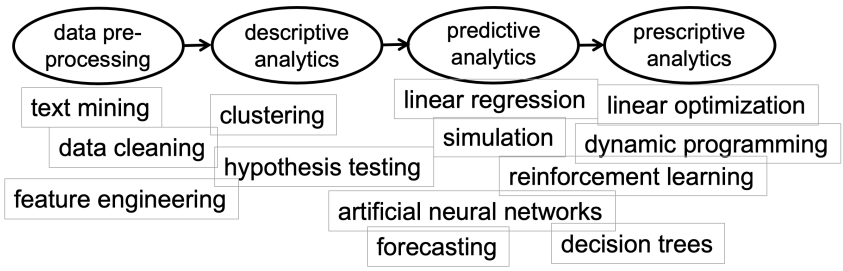



Figure 1.1: An overview of the most-often used data science techniques

Most predictive techniques require that you first structure the data. For example, topics can be extracted from text entered on social media or types of objects can be extracted from images. This brings us to a first distinction: between *structured* and *unstructured* data. Structured data usually consists of entries (e.g., people) with attributes (e.g., name, income, sex, nationality). The possible value for the attributes are well-defined (e.g., numerical, M/F, standard country codes). Structured data can be represented as a *matrix*: the rows are the entries, the columns the attributes.

Structured data comes in different flavors: for example, it can be numerical (e.g., temperature), categorical (e.g., days of the week), binary (e.g., true/false). Depending on the type of data different algorithms or adaptations of algorithms are used. If we have *univariate* data, i.e., data with only one attribute, then we can look at the distribution or compare different data sets. For *multivariate* data we can study how the different attributes influence each other.

Unstructured data has no such structure. It might be data from cameras, social-media sites, text entered in free text fields, etc. Counted in bytes, unstructured data is the majority of the data that is stored today, and it is often also big data. However, most of the BA and DS projects involve structured data, on which we will focus. When working with unstructured data, the first step is often to extract *features* to make it structured and therefore suitable as input for an algorithm working with structured data (e.g., images from road-side cameras are used to extract license plates which are then used to analyze the movement of cars).

Dealing with unstructured data is an important part of the pre-processing step. Cleansing is another one. Data often contains impossible values or empty fields. Different techniques exist to deal with these. A final important

pre-processing activity is *feature engineering*, combining attributes or *features* into new potentially more useful attributes. For example, combining "day of week" and "time" can lead to an attribute "business hours", and postal codes of individuals combined with census data can lead to an approximation of income and family composition.

Next we explore the data in the descriptive step. Typical activities are visualisation, different statistical techniques such as *hypothesis testing*, and *clustering*. Visualization is a technique as old as humanity, but it has developed tremendously over the last decades. Exploratory statistics will be discussed in Chapter 3. In clustering, you look for data points that are in some mathematical sense close together. Think about clustering individuals based in income, sex, age and family composition for marketing purposes.

In the descriptive step we do not focus on a target value (such as sales or number of patients cured). Having a target value is the defining distinction of predictive analytics. Therefore predictive analytics is also called *supervised learning*: we learn an algorithm to predict a target value based on a data set with known target values. In contrast, techniques such as clustering are considered *unsupervised learning*.

Supervised learning comes in two flavors: *regression* and *classification*. In regression we estimate a numerical value. The best-known methods are *linear regression* and *artificial neural networks* (which is actually a form of nonlinear regression), but other methods exist. In classification, the outcome is membership of two or more classes, e.g., whether or not somebody will click on an online ad, or vote on one of a number of parties. Most methods for regression can be adapted such that they can classify as well. Machine learning covers both supervised and unsupervised learning.

**Box 1.2. Human versus artificial intelligence**

Certain AI techniques are inspired by human intelligence or structures we find in nature, illustrated by names such as *artificial neural networks* or *evolutionary computing*. It is an interesting question whether or not we should try to copy human behavior with, eventually, the possibility that computers become "more intelligent" than humans. We could also argue that humans and computers have different capacities (seeing structures versus fast and errorless computation) and that our approaches to solving the same problem should be completely different. Your point of view might influences whether or not you find AI dangerous, as Stephen Hawkins did for example.

Often the set of known data entries is split in a *training* and a *test set*: the algorithm is *trained* on the basis of the training set, and then evaluated on

the basis of the test set. Usually an algorithm performs worse on the test set, but this is a more reliable comparison, as it avoids *overfitting*: the fact that the prediction of the algorithm is perfect for the training set but has no predictive value and therefore works bad on the test set. In statistics the terms *in sample* and *out of sample* are used for the same concepts. Understanding the background of the techniques and learning how to use them in the data science tool R is one of the main objectives of this book.

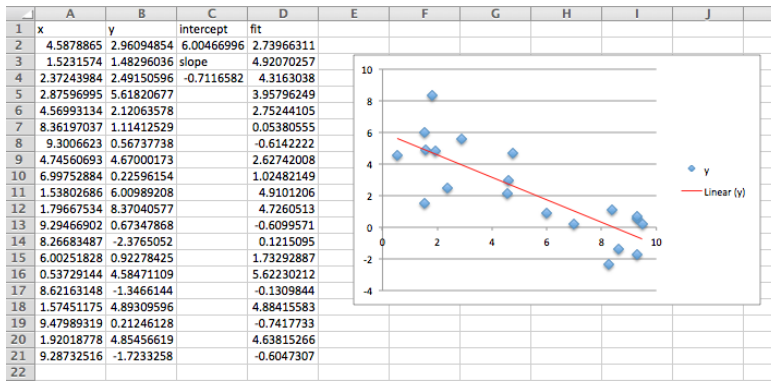
Descriptive analytics is *deductive* in nature: from the data set, we derive characterizing quantities such as means and correlations. Extending the knowledge from the training data to the whole population is *induction*. This is what we do in statistics and machine learning as part of predictive analytics. Certain predictive models combine deduction and induction: A real-life *system* is *modeled* using components. By predicting the behavior of the components (induction) we can deduce the behavior of the whole system. For example, in this way a production plant or the progression of a disease in a body can be *simulated*. By changing (the behavior of) certain components different scenarios can be analysed, leading to *optimization*, i.e., prescriptive analytics. Optimization comes in different flavors. Linear optimization is a powerful framework, used in many planning problems, such as crew scheduling in airlines and logistics. When problems are *dynamic* (e.g., they evolve over time, such as managing an investment portfolio), then *dynamic programming* is the right framework. When dynamic optimization is combined with learning, then we speak of *reinforcement learning*.

## 1.4 Tooling

A multitude of tools exist to assist the data analyst with his or her task. We first make a rough division between *ad hoc* and *routine* tasks. For routine tasks, standardized and often automated procedures exists for the process steps, often involving dedicated and sometimes even tailor-made software. For example:
- for data collection *data warehouses* exist with connections with operational IT systems;
- for distribution companies *decision support systems* exist that compute the optimal route of delivery trucks, saving many transit hours and petrol.

We will first go into detail on software for ad-hoc tasks. For ad-hoc tasks there are a number of proprietary and open source tools—*R* (open source) and *MS Excel* (proprietary) are among the most popular ones. Both allow

the user to efficiently manipulate data, often represented as matrices. Each tool functions in very different ways: R manipulates data in a declarative way, very much like programming languages. Additionally, the interactive environment *RStudio* allows for an easy manipulation of data, R scripts and figures. Excel is essentially a 2-dimensional worksheet, with the possibility to perform calculations in each cell and to add entities such as figures. Both have many useful functions, for example for statistical calculation. Many libraries exist containing algorithms that can be added to these tools, both open source and proprietary. Users can also add new functions or libraries to both tools. For a screenshot of a simple implementation of linear regression in both R and Excel, see Figures 1.2 and 1.3.



| | A | B | C | D |
|---|---|---|---|---|
| 1 | x | y | intercept | fit |
| 2 | 4.5878865 | 2.96094854 | 6.00466996 | 2.73966311 |
| 3 | 1.5231574 | 1.48296036 | slope | 4.92070257 |
| 4 | 2.37243984 | 2.49150596 | -0.7116582 | 4.3163038 |
| 5 | 2.87596995 | 5.61820677 | | 3.95796249 |
| 6 | 4.56993134 | 2.12063578 | | 2.75244105 |
| 7 | 8.36197037 | 1.11412529 | | 0.05380555 |
| 8 | 9.3006623 | 0.56737738 | | -0.6142222 |
| 9 | 4.74560693 | 4.67000173 | | 2.62742008 |
| 10 | 6.99752884 | 0.22596154 | | 1.02482149 |
| 11 | 1.53802686 | 6.00989208 | | 4.9101206 |
| 12 | 1.79667534 | 8.37040577 | | 4.7260513 |
| 13 | 9.29466902 | 0.67347868 | | -0.6099571 |
| 14 | 8.26683487 | -2.3765052 | | 0.1215095 |
| 15 | 6.00251828 | 0.92278425 | | 1.73292887 |
| 16 | 0.53729144 | 4.58471109 | | 5.62230212 |
| 17 | 8.62163148 | -1.3466144 | | -0.1309844 |
| 18 | 1.57451175 | 4.89309596 | | 4.88415583 |
| 19 | 9.47989319 | 0.21246128 | | -0.7417733 |
| 20 | 1.92018778 | 4.85456619 | | 4.63815266 |
| 21 | 9.28732516 | -1.7233258 | | -0.6047307 |
| 22 | | | | |



Figure 1.2: A simple implementation of linear regression in Excel

Excel is omnipresent and is easy to learn, making it the favorite tool for many people doing relatively easy computational tasks. However, Excel is also known for the errors users make with it. This might be partly due to a lack of appropriate training, but the lack of structure also contributes. R enforces more structure, just like programming languages do. Learning R requires more time but it may well be worth the investment.

As mentioned, Excel and R are both analytic environments with many built-in functions. Engines can be called from these environments to perform certain tasks, such as optimization. These engines can also be proprietary or open source. For example, for *linear optimization* (see Chapter 6), the best solvers, Gurobi and CPLEX, are proprietary; CBC is an example of an open-source solver.

Many other environments exist, often for specific analytics tasks. Examples are SPSS, often used in social sciences for statistical analysis, and

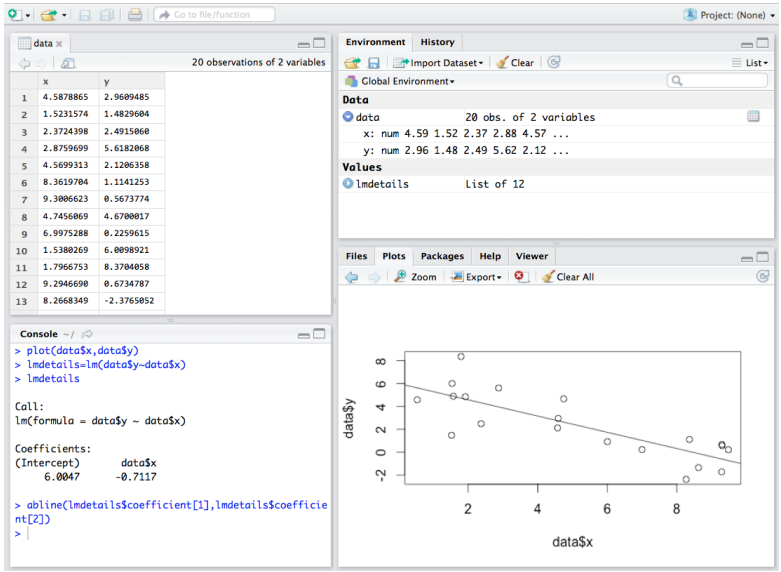



Figure 1.3: A simple implementation of linear regression in RStudio

AIMMS, an optimization environment. A special place is taken by *Python*. Python is a programming language with libraries containing many functions for data analysis. For this reason it is both used for ad hoc analysis and routine tasks, solving the so-called "two-language problem", the fact that you have to move from say R to a language like Java or C++ once you move from a successful pilot to a production system.

Excel, although very different in functionality, is also often used for routine tasks. It has some functionality for this, such as the possibility to connect to databases and to add user-friendly screens, but it lacks others, such as user management. Although in principle everything can be built within Excel, thanks to the underlying programming language VBA (*Visual Basic for Applications*), in practice it often leads to slow error-prone systems consisting of a spaghetti of multiple sheets referring to each other.

Concerning software for routine tasks, there is a large variety in possible tooling. A major difference is between *off-the-shelf* and *tailor-made* software. In the area of prescriptive analytics *decision support systems* (DSS) form the main category of off-the-shelf software. This is software built for a specific goal, such as the routing of delivery vans or the pricing of hotel rooms. Next to the analytics algorithms, DSS typically have built-in connections to data sources and allow the user to interact with the software in such a way that

input and output of the algorithms can be manipulated.

In the area of data collection and descriptive analytics BI tools exist, such as IBM Cognos, that help the user collect data and execute queries. Recently, many tools are built to store and manipulate big data. Google and Amazon are major players in this area with the open-source database and data manipulation systems *Hadoop* and *Mapreduce* (mainly developed by Google) and *Amazon Web Services*, providing big data cloud storage and computing.

Tailor-made analytics software can be written in many different languages. We already mentioned Python, but popular languages include php, Java, C++ and C#, combined with mySQL (open source) or MS SQL server databases. Note the move of proprietary off-the-shelf tooling to cloud-based solutions, taking away the need for expensive servers at the customer site, and making maintenance and support much easier.

In Figure 1.4 you can find an overview of the tools discussed. In this book we will mainly use R.

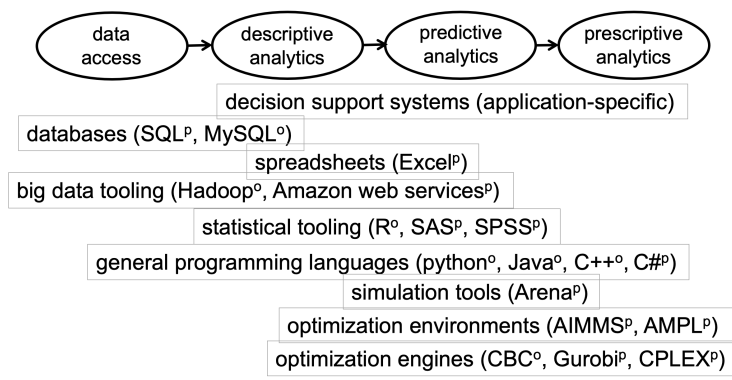



Figure 1.4: Types of analytics tools with some examples;
o = open source, p = proprietary

Note that many interfaces exist between the tools and languages in Figure 1.4. From within Excel and general programming languages databases can be accessed; DSS, spreadsheets and optimization environments call optimization engines, etc. Especially with the open source environments R and Python every imaginable data science project can be done, where python is preferred in the case of big data or applications requiring intensive computation. R and python are quickly gaining popularity: there is an enormous community developing new libraries and offering support through websites such as `stackoverflow.com`.

## 1.5 Implementation

A successful implementation of BA requires the right combination of tools and skills from the BA consultant(s). But more is needed: the organization should have reached the right maturity level to make the implementation possible. Let us consider first the required skills of the specialist.

The core knowledge of any BA specialist is the command of suitable tooling (such as R) and a broad understanding of descriptive, predictive and prescriptive methods. Next to that, a specialist might have management skills (project management, change management, communication skills), programming skills (in for example C++ or Python), or deep knowledge on some of the technical areas, often clustered by the scientific disciplines of statistics, machine learning or optimization. These specialists are considered to be "T-shaped": they have breadth and also depth in a certain area. Sometimes people talk even of "Π-shaped", emphasizing the importance of knowledge of the application domain, the second vertical bar. However, the importance of breadth cannot be underestimated: It is important to be able to use the right method for the problems one encounters. Scientists are still too often specialized in one tool (e.g., a hammer) which they use for all problems they encounter (e.g., to put a screw in the wall).

It is crucial to have good analysts, but an organization should also support the deployment of analytics. The extent to which an organization supports a certain concept is called its *maturity* with respect to this concept. The maturity is measured using *maturity models*. Different analytics maturity models have been developed. The more mature an organization, the higher the impact of analytics. We illustrate the concept using the "INFORMS Analytics Maturity Model" [1]. It consists of three sets of questions, concerning the organization, its analytics capability, and its data and infrastructure. On the basis of this a score is calculated. For example, an organization with a central data warehouse and a centralized analytics strategy will score higher than a company lacking these.

## 1.6 Additional reading

General information on many subjects can be found on Wikipedia. We already mentioned Davenport & Harris [8], which is still an interesting non-technical book to read on the value of BA.

For more background on errors in Excel see Powell et al. [29] and other

**Box 1.3. Legal and ethical aspects**

Although not his or her main focus, a data scientist should be aware of legal and ethical aspects. The legal aspects often start with the data collection: are you allowed to get and analyze the data? Some form of *data anonymization* can be useful in this process. Current laws (such as the EU GDPR regulation) also limit the amount of time you are allowed to keep data, which contradicts the wish to keep as much data as possible for future analysis.

There are many privacy issues that have to do with data, like who has access to data about you, who owns it, and how do you know which data is out there about you? Ethical questions also arise around the use of algorithms. On what basis do algorithms make decisions about for example employment? Algorithms can be discriminating because they were trained to do so by the data. On the other hand, a data science approach can also give solutions, for example by communicating all parameters of a predictive model.

papers by the same authors.

Some interesting ideas on the different profiles of data scientists (on which part of Section 1.5 is based) can be found in Harris et al. [14].

More information on project management can be found in Klastorin [21]. A classic on change management is Kotter [22].

You can try the INFORMS Analytics Maturity Model yourself at [1].

A well-know mathematician and author writing on ethical issues of data science is Cathy O'Neil, see for example her TED talks and [27].

# Chapter 4

# Machine Learning

In this chapter we focus on models for *multivariate* data. For every entry in the data, we have multiple fields or *attributes*. First we look at *clustering*: can we find groups of data points that are clearly separated from each other? Then we move to models where each entry has a target value that we should explain from the other fields. This target value can be real (any number) or 0/1. In the first case, linear regression (LR) and *deep learning* (a type of *artificial neural network* or ANN) are popular methods. For the 0/1 target we will see how LR and ANN can be adapted to deal with this case. We will also look at *support-vector machines* and *decision trees*. *Forecasting* is a special case of estimation in the situation where one of the fields contains time stamps. We dedicate a separate section to this scenario.

This chapter is entitled *machine learning* (ML), because this name well describes the ideas found within this chapter: using algorithms we discover the values of parameters that describe the structure of the data. This chapter covers the main methods of the field that is nowadays called ML. Note however that ML is a subset of the older field, referred to as *data mining*, and that there is a large overlap with statistics. The difference between ML and statistics has less to do with the methods than with the approach. Statisticians have a more mathematical approach, focusing for example on showing that an algorithm always finds the best parameter values. People with an AI background tend to focus more on computational results. Also, statistics is more focused on small datasets, while typical ML methods need larger datasets (usually a few thousands of observations) to work.

**Learning outcomes** On completion of this chapter, you will be able to:

- perform simple machine learning tasks in R
- choose an appropriate method in a given situation
- translate an prediction problem in practice to a machine learning problem
- reflect on the usefulness of machine learning and its pitfalls

## 4.1 Data preparation

A large proportion of the time devoted to an ML project goes to the preparation of the dataset before the actual ML technique can be executed. In this section we discuss this *data preparation* step. Of course, data preparation is also relevant for univariate data, but because many of the techniques for data preparation concern multivariate data, we discuss it here.

Data preparation consists of the following activities: *data cleansing* (or *cleaning*) and *feature engineering*. *Data wrangling* and *data pre-processing* can be seen as synonyms of data preparation. Especially data wrangling is a more recent term that came into existence with the increase in attention paid to data preparation. Also an additional job title emerged: the *data wrangler*. Another recent job description, with less focus on the algorithms and more on the business side, is *citizen data scientist*.

Regarding data cleansing, data often has missing fields in part of the entries. Many algorithms require all fields to be filled in, often leading to the removal of incomplete records, or to the use of methods such as *mean imputation*, which consists of replacing missing data with the mean of the existing data.

Something else that often occurs are incorrect values, for example negative values for age. Removal or replacement by the average are again possibilities. A better solution however, is to improve the data collection process to avoid missing fields and incorrect values altogether. Indeed, analytics projects lead to more awareness of the importance of data collection and in the end also result in better data. For this reason, in many cases only recent data can be used, sometimes leading to a shortage of data and the risk of *overfitting*, which is the case when a model fits the data well but has no predictive value. This will be discussed in Section 4.4.

**Exercise 4.1** *Consider the* `airquality` *data set of the* `datasets` *library. The* `Solar.R` *attribute contains quite a number of missing values indicated with NA*

*("not available"). Determine the mean of the non-NA values and replace the NA values by this number.*

Data cleansing requires knowledge of the area of business to which the data relates to be able to understand whether or not data is incorrect. The next step in data preparation—feature engineering—also requires domain knowledge.

Feature engineering consists of different steps. Certain attributes have to be aggregated into appropriate *bins*. For example, the minutes of the day can be aggregated into 24 bins, one for each hour, or even 4: morning, afternoon, evening, night. This is called *binning*. Choices have to be made: do we aggregate income data into "low", "medium" and "high"? Or do we keep the numerical value? This depends on the relation with other attributes, especially the target attribute, and the type of algorithm that will be used.

An important step is the aggregation of multiple attributes into new attributes or *features*, hence the name feature engineering. This often requires inside knowledge of the application area. For example, date and time can be combined into a binary attribute "business hours", when this might be relevant for the problem being studied (e.g., bookings for hotels or flights). Visualisation and insight how attributes correlate are important parts of deciding which attributes to construct and to add. It is not always the case that you first prepare the dataset and then run the ML algorithm once; often you go back and forth between feature selection and learning.

ML methods require structured data as input. For this reason unstructured data such as images and free text have to be transformed into structured data. This requires image recognition and text analytics.

Some methods (such as neural networks) perform better when data is centered and scaled. A standard way to do this is by replacing value $x$ by $(x - \mu)/\sigma$ with $\mu$ the average attribute value and $\sigma$ its standard deviation.

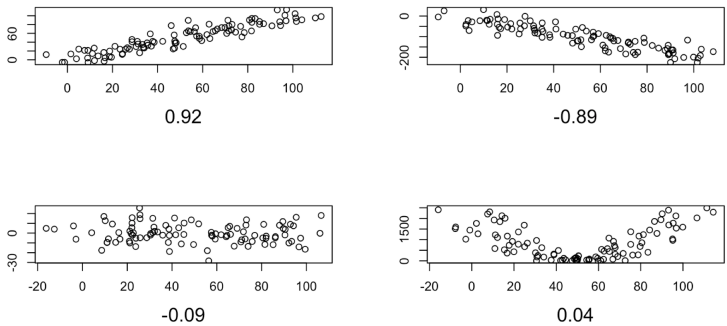
## 4.2 Clustering

Clustering is the only unsupervised learning technique that we will discuss. Unsupervised learning means there is no target value that we have to approximate or predict. For example, in hotel reservation data we see two separate clusters with different behavior (days of the week, price, time to book), but there is no target attribute with the true value (as in *classification*).

Many different clustering methods exist, depending on the types of data and the objective of the clustering. Here we only describe the most common

**Box 4.1. Correlation coefficient**

The correlation coefficient, or Pearson correlation coefficient, measures the extent to which two variables have a linear relation. It ranges between $-1$ and 1, with $-1$ meaning perfect negative correlation, 0 no linear correlation and 1 a perfect positive correlation. Examples with the coefficients are given below.




A relation such as in the right-bottom figure suggests a non-linear relation, for example a parabola. In those cases, it is useful to add non-linear attributes, for example $x^2$, especially when linear regression is used.

**Box 4.2. Text mining and image recognition**

For unstructured data to be useful in machine learning, features have first to be extracted. This holds true for both images and text. Text mining (today often referred to as *text analytics*) in its simplest form entails looking for keywords in free text fields. More advanced techniques include some form of understanding of text using concepts from linguistics and *information retrieval* (which is the technique behind search engines, which is also about extracting knowledge from free text).

However, these techniques are not always successful in extracting all relevant information out of free text. For this reason, for ML purposes, it is often better to have a limited choice with, for example, a drop-down menu instead of the possibility to enter any text.

Also, image recognition involves the extraction of features from unstructured data, in this case patterns in a 2-dimensional data set with color codes for all pixels. These features are then used as input for ML algorithms, typically *neural networks* with multiple layers (which is called *deep learning* for this reason).

one: $k$-means clustering, used for numerical data. An important property is that the number of clusters found is fixed from the beginning, at say $k$. We assume we have $d$ numerical attributes, thus every record is a point in the $d$-dimensional Euclidian space. The algorithm works as follows:

*k*-MEANS CLUSTERING

| | |
|---|---|
| **initialization** | select randomly $k$ points in the $d$-dimensional Euclidian space |
| **assignment** | determine for each record the Euclidian distance to all $k$ points and assign it to the closest |
| **update** | move the $k$ points to the *centroid* of each cluster by taking component-wise averages |
| **repeat** | repeat the assignment and update steps until the clusters remain the same |

See Figure 2.3 on page 25 for an example and for relevant R code.

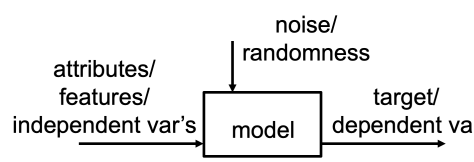
**Exercise 4.2** *Consider the* `rock` *data set of the* `datasets` *library. Use k-means to determine 3 clusters and put the centers together with the data in a scatter plot showing* `area` *and* `peri`*. Does this correspond to what you expected?*

## 4.3 Linear regression

Next we study linear regression. This is an important step, because now we move from descriptive methods such as looking at correlations and clustering to predictive methods such as linear regression and neural networks.

In predictive models, there is some *dependent variable* that we want to predict using other variables, the *independent variables*. A dataset with values for the dependent and independent variables is available to *train* or *fit* the model. After that it can be used to predict the dependent variable for points for which we know only the independent variables.

Unfortunately, the outcome is not fully determined by the independent variables: there is also *noise*, which is an unknown component. Therefore, the predictions are never exact. Obviously, it is our goal to get as close as possible.




On the other hand, although it may sound counterintuitive, it is sometimes a bad sign when the *fit* is (almost) exact. Certain models have a good

fit but they are bad predictors. The reason is that they fit a coincidental pattern in the noise. The same pattern is unlikely to occur for the values to predict, therefore they predict badly. This phenomenon is called *overfitting*.

**Box 4.3. Fit versus prediction**

The fit is defined as the outcomes of the ML model for the points for which we know the target. The prediction is for points for which we do not know the target value. In the case of temporal data (such as sales per day), we call the prediction the *forecast*. Thus, the fit concerns the past and the forecast the future. (For more on forecasting, see Section 4.5.)

**Example 4.1** *Height is strongly correlated to weight of people, therefore height is a good attribute to predict weight. In a particular dataset however, it might be the case that, coincidentally, people born on Wednesday weigh more than those born on other days. Introducing day of the week in our predictive model is an example of overfitting: when we predict the weight of a new group of people, we overestimate for those born on Wednesday, because it is unlikely that they are also heavier. (Although we can never exclude a real correlation: e.g., perhaps the local hospital plans all caesarians for overweight babies on Wednesday.)*

**Box 4.4. Overfitting**

When the number of variables in a model is not in balance with the number of data points then overfitting can occur. To test for overfitting, it is customary to split the data into a *training set* and a *test set*. Then the model is fit to the training set and it is *validated* on the test set. When this procedure is repeated multiple times on different subsets of the data, then we call this *cross-validation*. This is common practice in ML projects. A rule of thumb is that you should have at least 10 data points per attribute, known as the *one in ten rule*.

In statistics, the terms *in-sample* and *out-of-sample* are used. For example, the in-sample error corresponds to the difference between the model and the actual value on the training set.

A related concept is the *bias-variance trade-off*: when the bias (the difference between signal and fit) is high you miss the signal (*underfitting*); a high variance (of the fit) is an indication of overfitting. While there is a trade-off, a highly predictive model has both a low bias and a low variance.

Linear regression is the most commonly used predictive method.

We approximate or estimate each value $x = (x_1, \ldots, x_d)$ in the $d$-dimensional attribute space with:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d.$$

The parameters $\beta_0, \ldots, \beta_d$ are determined using a (training) set of known values, with $n$ records, with record $i$ denoted as $(x_{i1}, \ldots, x_{id}, y_i)$. The objective is to get $\hat{y}_i$ and $y_i$ as close as possible in the following way: we minimize the *sum of squared errors* (SSE)

$$\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_d x_{id} - y_i)^2.$$

There is an algorithm that finds these $\beta$s, called the *least-squares algorithm*. In R you can use it as follows, with the first line omitting all rows with `NA`s:

```
> aq=na.omit(airquality)
> summary(lm(Ozone~.,data=aq))
```

`lm` stands for *linear model*, the attribute left of the $\sim$ is the target, "." right of it indicates all other attributes. Let's have a closer look at the output, given below.

```
Residuals:
    Min      1Q  Median      3Q     Max
-37.014 -12.284  -3.302   8.454  95.348

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -64.11632   23.48249  -2.730  0.00742 **
Solar.R       0.05027    0.02342   2.147  0.03411 *
Wind         -3.31844    0.64451  -5.149 1.23e-06 ***
Temp          1.89579    0.27389   6.922 3.66e-10 ***
Month        -3.03996    1.51346  -2.009  0.04714 *
Day           0.27388    0.22967   1.192  0.23576
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.86 on 105 degrees of freedom
Multiple R-squared:  0.6249,    Adjusted R-squared:  0.6071
F-statistic: 34.99 on 5 and 105 DF,  p-value: < 2.2e-16
```

We focus on three values: the estimates of the $\beta$s, the $p$-values of each attribute, and the $R^2$. From the $\beta$s, we see the impact of each attribute on the Ozone level: for example, if the temperature increases by 1 degree, then the Ozone level increases by 1.89. To the right are the $p$-values. Only `Day` is above 5% and thus insignificant. Therefore we reduce the model to:

```
> summary(lm(Ozone~Solar.R+Wind+Temp+Month,data=aq))
```

Now `Solar.R` is not significant, and thus we take:

```
> summary(lm(Ozone~Wind+Temp+Month,data=aq))
```

Now all attributes are significant and we arrive at our final model. This is called the *step-down* approach. Its goal is to separate noise and signal:

statistically speaking it is very unlikely that the dependence on `Wind`, `Temp` and `Month` is due to noise.

**Box 4.5. Regularization**

Other more general methods to prevent overfitting are known under the name *regularization*. These methods add a penalty term on the regression parameters to the least squares formulation, reducing the number of non-zero parameters and their values. This penalty can be linear (LASSO), quadratic (ridge regression), or a combination of both (elastic net). Applied to the `airquality` data this leads to models with only `Temp`, and then, as the weight of the penalty is decreased, `Wind`, `Solar.R` and `Month` are added. The best weight should be determined using cross-validation.

Some R code to get started:

```
> library(glmnet)
> lasso=glmnet(model.matrix(Ozone~.,aq)[,-1],aq[,1],alpha=1)
> predict(lasso,type="coefficients")
```

Therefore, we now have a significant model without overfitting. But is the prediction good? The most common measure compares the SSE with the sum of squared errors of the most naive predictor: the mean. The *coefficient of determination* $R^2$, which can also be interpreted as the fraction of explained variance, is given by:

$$R^2 = 1 - \frac{\text{SSE of regression}}{\text{SSE of mean}}.$$

$R^2$ is always between 0 and 1. The closer to 1, the more of the variance is explained by the regression. In the `airquality` example it is 62%, which is not bad, although 38% of the variability remains unexplained.

The ultimate goal of predictive modeling is to predict the dependent variable for new cases. We predict row 10 of the `airquality` data set which has NA as Ozone value. The R command:

```
> predict(lm(Ozone~Wind+Temp+Month,data=aq),airquality[10,])
```

gives prediction 36.57.

**Exercise 4.3** *Consider the* `longley` *dataset. Use the step-down method to build a significant model for* `Employed`*. What is the R squared? Use the model to predict the target for all mean attribute values.*

So far, we discussed LR in its basic form. However, the method is more versatile than the name suggests: the linearity is related to the $\beta$s, not to

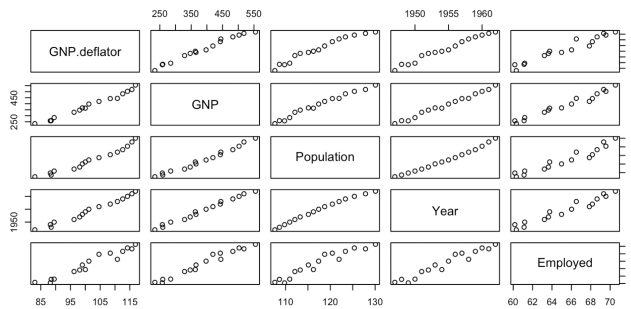
**Box 4.6. Causality**

Statistical correlation does not necessarily imply *causality*. It regularly happens that there is another hidden variable, called a *confounding variable*, that causes both. Therefore you would need additional evidence to make conclusions concerning causality.

For example, it appears that dog owners are less stressed. This does not necessarily mean that getting a dog will reduce your stress: perhaps people having more time are less stressed and more likely to get a dog. If you give already stressed people the responsibility of a dog, they might even get more stressed... A way to prove that dog ownership reduces stress would be to give dogs to a sufficiently large, randomly selected group of people not owning dogs and study the (long-term) consequences. This is a very costly, if not impossible, thing to do. A more practical approach is to add possible confounding variables to the model. For example, if you add "free time" to the model, the positive effect of dog ownership on stress might disappear.

**Box 4.7. Collinearity**

In Exercise 4.3 we found a number of independent variables with a very strong linear relation, see below.




All these variables have approximately the same predictive value (which is also high because of the linear relation with the dependent variable `Employed`). To avoid ambiguity it is better to select one. The choice of variable might also be influenced by causality: it might be better to use `GDP` or `Population` than `Year`. Note that to avoid collinearity it might be interesting to introduce a new variable `GNP per capita`. In the current case the dataset is too small to get significant results.

the independent variables. For example, by replacing `y~x` by `y~poly(x,2)` you get a quadratic relation (a 2nd-order polynomial).

**Exercise 4.4** *Make a data frame with a quadratic function, for example by* `> df=data.frame(x=1:20,y=3-(1:20)+0.5*(1:20)^2)`.

*Find a perfect fit using* `lm`.

Next to adding other functions of a single variable, such as polynomials, we can also introduce functions of multiple variables, such as $x_1x_2$. These are called *interactions*. You automatically get all interactions if you replace "+" by "$*$" in the call of `lm`, for example $y \sim x_1 * x_2$ instead of $y \sim x_1 + x_2$.

**Exercise 4.5** *Try to improve the results for the* `airquality` *dataset by adding one or more interactions.*

Another important feature are categorical independent variables. If such a variable can take only 2 values, e.g., 0 and 1, then it fits immediately within the framework. But also in the case of multiple values, it is possible to use linear regression, by adding a variable per value. R does this automatically, except when the values are numerical. An example is the `Month` attribute in the `airquality` data frame: it takes values 5–9, which is different from taking May–Sept. When looking at the data, for example by `pairs(airquality)`, we see that the relation between `Month` and `Ozone` is far from linear. This can be solved by making `Month` a *factor*, leading to:

```
> summary(lm(Ozone~Wind+Temp+factor(Month),data=aq))
```

We see that R squared has increased a bit. The method of adding a variable per value for categorical variables is called *one-hot encoding*, the variables are known as *dummy variables*. The statistical theory is known under the name *Analysis of Variation* or ANOVA.

**Exercise 4.6** *Analyse the influence of supplement and dose on tooth growth in the dataset* `ToothGrowth`. *Model both variables as categorical. What is the best combination of supplement and dose?*

**Box 4.8. Regression towards the mean**

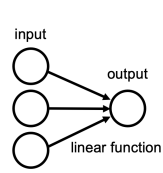
The word "regression" in linear regression comes from the concept of *regression towards the mean*: the fact that after an extreme data point the next one is more likely to be close to the mean. "Linear prediction" would probably have been a more accurate term than linear regression.

## 4.4 Nonlinear prediction

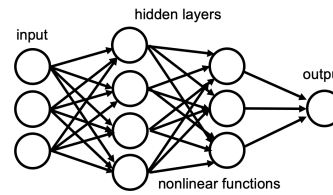
Although linear regression is quite versatile, we are restricted to functions that are linear in the $\beta$s. That is, we can model $\beta x$, $\beta x^2$, and even $\beta x_1 x_2$, but

not functions like $x^\beta$ or $\mathbb{I}\{x \leq \beta\}$ (which is 1 for $x \leq \beta$ and 0 otherwise). Note that, just as in linear regression, $x$ is here the attribute value and $\beta$ is the parameter of the predictive model. To be able to capture these types of relations we have to consider nonlinear predictive methods, which is the subject of this section. We will discuss *artificial neural networks* and methods originating from *decision trees*.

In LR the output is a linear function of the input, see the formula on page 61 or the figure below. An ANN is a network of nonlinear functions connecting input and output. These functions are of a special form: the output of each node is a nonlinear function such as $\max(0, x)$ with $x$ the weighted sum of the inputs to that node.




linear regression



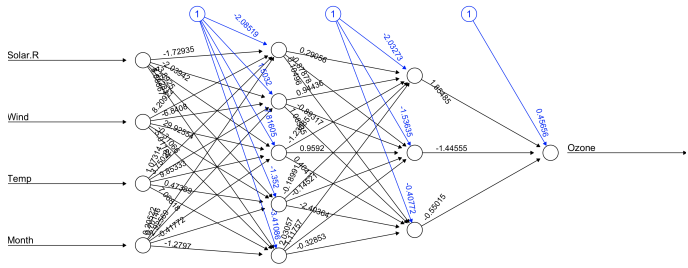

artificial neural network

Each arrow in the network has a parameter, and training of the network consists in finding the parameters that minimize the error on the outputs. The algorithm to do this is more complicated and time-consuming than in the case of linear regression, especially when the network has many layers and nodes per layer. This iterative algorithm is called the *backpropagation* algorithm.

**Example 4.2** *We will apply the* `neuralnet` *package to the* `airquality` *data. An important pre-processing step is to scale all data to numbers between 0 and 1. The R-code is as follows:*

```
> library(neuralnet)
> maxima = apply(aq, 2, max); minima = apply(aq, 2, min)
> aq_scaled = as.data.frame(scale(aq, center = minima,
        scale = maxima - minima))
> nn = neuralnet(Ozone~Solar.R+Wind+Temp+Month,data=aq_scaled,
        hidden=c(5,3))
```

*This leads to the following neural net:*




*As an example of its use, the fit can be calculated as follows:*

```
> fit = compute(nn,aq_scaled[,2:5])$net.result*
        (maxima[1]-minima[1])+minima[1]
```

*Initial weights are set randomly, therefore the outcomes may differ. The resulting R squared is typically around 85%.*

ANNs have many parameters, especially when multiple hidden layers are used. To avoid overfitting, we should always work with a training and a test set, and the net should be trained with as many examples as possible.

**Exercise 4.7** *Compute the R squared of the* `airquality` *example. Compare it to the R squared of the linear model. Split the data in a training and a test set, and compute the R squared on the test set using both LR and ANN.*

**Exercise 4.8** *Consider the* `iris` *dataset. Use R to train a neural network that predicts* `Sepal.Length` *based on the other attributes. Use it to predict a case of* `setosa` *with average* `Sepal.Width, Petal.Length` *and* `Petal.Width`*.*

**Box 4.9. Activation functions**

The history of ANNs goes back to the 1950s. At the time, there was not sufficient computational power to execute the backpropagation algorithm on sufficiently large networks. Over the last decades, this situation has significantly changed, especially since a new *activation function*, the so-called *ReLU*, is used. Traditionally, the nonlinear function in every node is taken to be the *logistic function* $e^x/(1+e^x)$. This function has many desirable mathematical properties, but an important step in making *deep learning* (ANNs with many hidden layers) work, was the use of the rectified linear unit (ReLU), given by $x^+ = \max(0, x)$.

The use of ANNs with many hidden layers is called *deep learning*. This technique has been very successful in many prediction tasks over the last

decades, especially for *image* and *speech recognition*. Deep learning is the technique behind all developments related to autonomous driving. A lot of the technology is publicly available: for example, the R keras library gives an interface to Google's open source library for deep learning *Tensorflow*. Deep learning needs lots of computing power requiring special equipment, such as GPUs (*graphics processing unit*) of which the different cores, used to command different parts of the screen, can now be used to execute back-propagation in parallel. Another aspect of deep learning is that it needs to be configured in the right way, for example by feature engineering. For problems with structured data, a method such as gradient boosting (to be discussed next) works in general better, and it requires little configuration.

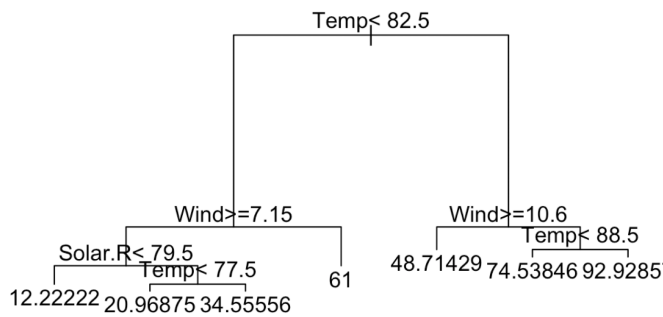
**Box 4.10. Explainable data science**

A big disadvantage of ANNs is that it is hard to understand the model: it is a *black box*, you do not know what is happening inside. In linear regression, the parameters tell us what the impact is of each independent variable, whereas the multitude of parameters in neural nets give us no clue about the impact of each variable. This is a serious drawback of ANNs: decision makers in general want to understand why they need to make a certain decision, not just because the algorithm "says so". There are also situations where black-box predictions lead to ethical questions.

Let us build a *decision tree* for the `airquality` data. The following code prints a tree as text:

```
> library(rpart)
> rpart(Ozone~.,data=aq)
```

A graphical representation of the same tree is given below.




It should be interpreted as follows: for a certain data point, you start at the top, and then, based on the outcomes of the logical statements, you go left for true and right for false until you hit a leaf with a number, which

is the prediction for that data point. For example, a day has Temp 80 and Wind 5. Then the prediction is 61: because Temps is smaller than 82.5 you go left, and then right because `Wind>=7.15` is false.

Branches are made on the basis of which logical expression has the highest predictive power, i.e., which one decreases the most the sum of squared errors. This is done iteratively until some stopping criterion is reached.

**Exercise 4.9** *Determine the quality of the decision tree for the* airquality *data in the same way as for neural nets.*

Decision trees have several advantages: they are easy to make, also for large data sets, and they are easy to interpret. However, their performance is not very good unless trees with many layers are created, which have the risk of overfitting. There exist extensions, called *ensemble methods*, that use multiple trees and decrease the risk of overfitting. There are two methods: *bagging* and *boosting*.

Bagging is a general method by which multiple, say $B$ predictors are constructed, each using a different training set based on the original one. These $B$ training sets are constructed from the original one by random selection with replacement. For each training set a decision tree is constructed. The final predictor is the average over the $B$ predictors. Compared to a single decision tree, this predictor typically has a lower variance and the same bias. We call this a *bagged tree*. We can further improve the method by selecting the features randomly. This avoids strong correlations between the trees. The resulting set of trees is called a *random forest*.

**Exercise 4.10** *Use the* randomForest *R package to create a random forest predictor for the* airquality *data. Determine its performance again using a training and test set.*

It is of interest to note that bagging different methods, i.e., combining different regression methods (such as linear regression and decision trees), works very well in general, because the strong points of the different methods are combined.

*Gradient boosting* is another ensemble method often used with decision trees. It only assumes you have a prediction method that is just better than taking averages (a so-called *weak learner*). Now you iteratively build up your prediction method: the error of the previous step is (partially) corrected by a new tree. The error of this new set of trees is corrected by yet another

tree, etc. The interesting fact is that this procedure (called a *meta-algorithm* because it uses other algorithms as input) produces a *strong learner*: a prediction that reaches an arbitrary precision when applied to enough data.

Gradient boosted decision trees is one of the most popular and most successful ML methods today, winning many recent Kaggle competitions. Because it is based on trees, it gives (some) insight into the importance of attributes and it works well "out-of-the-box".

**Exercise 4.11** *Use the* xgboost *R package to create a gradient boosting predictor for the* airquality *data. Determine its performance again using a training and test set.*

**Exercise 4.12** *Consider the* `iris` *dataset. Use R to train a decision tree, a random forest and gradient boosting to predict* `Sepal.Length` *based on the other attributes. Use it to predict a case of* `setosa` *with average* `Sepal.Width, Petal.Length` *and* `Petal.Width`*.*

## 4.5 Forecasting

Forecasting is a special kind of prediction involving temporal data, i.e., data which is ordered by time. Examples are quarterly unemployment rates, daily sales of a product and heart-rate measurements made every second by a fitness tracker. In the scientific literature there is a strong bias towards long-term economic data, but today most of the data is automatically collected *high-frequency data*. Because it is a special case of prediction, the methods discussed in the previous section can sometimes be applied successfully. However, *time series* (data with time stamps) often have special features that make dedicated forecasting methods work better. Indeed, we often see seasonality (for example, the same patterns occuring every year), errors on consecutive days that depend on each other (violating standard assumptions), and the fact that older data is often less relevant. Over the last century many special-purpose algorithms have been developed, mainly by statisticians and econometrists. The most important ones are smoothing methods and ARIMA (see Box 4.12). Our focus will be on the highly successful smoothing methods.

Time series usually consist of three components: trend, seasonality, and noise. Trend is the long-term level. Seasonality concerns fixed-length cycles, usually of a year, week or day. Noise is random short-term fluctuations.

**Box 4.11. M-competitions**

Many different forecasting methods exist, but which one is best? A number of competitions have been organized by the forecasting specialist S. Makridakis, and therefore called the *M-competitions*. The most recent one at the writing of this book (M4) took place in the first half of 2018. A number of conclusions can be drawn:
- mathematical sophistication is not a guarantee for better forecasts;
- pure ML methods such as deep learning are less good than traditional statistical methods;
- combining (*bagging*) methods improves the forecasting accuracy.

**Box 4.12. ARIMA**

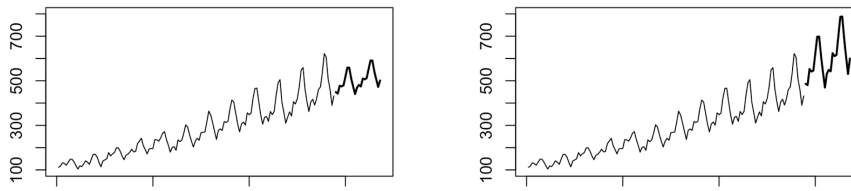
ARIMA stand for *auto-regressive integrative moving-average*, and is a mathematical model where the error and the parameters depend on previous errors and parameters. A mathematically beautiful theory, but a black box that usually performs worse than methods based on decomposition or smoothing. For this reason it is seldomly applied in practice.

**Example 4.3** *Sales data show all three components. Sales might increase or decrease over time depending on market share, competition, economic situation, etc. This is part of the trend. Sales of consumer articles often show a strong end-of-year peak: this is intra-year seasonality. Sales also depend on the day of the week and the hour of the day, which is the intra-week and intra-day seasonality. Finally, even the best forecasting algorithm cannot forecast with 100% accuracy: what remains is noise.*

An obvious way to forecast is to take time and seasonality (using one-hot encoding) as independent variables and use one of the methods of the previous sections.

**Example 4.4** *The* `AirPassengers` *data set contains 12 years of monthly data. By adding a variable with the names of the months, we model trend and seasonality at the same time. With the following code we can make a linear model, with a 2-year forecast as shown in the left plot:*

```
> time=1:144; months=rep(month.abb,12)
> fit=lm(AirPassengers~time+months)
> fc=predict(fit,data.frame(time=145:168,months=month.abb))
```




*The seasonality of the forecast is not big enough: it does not scale with the trend, which is what we could have expected from a linear model. By transforming the data first (by taking logarithms) and then later taking exponentials, of the forecast we get the better forecast on the right-hand side.*

As the objective used in forecasting we usually take the *root mean squared error* (RMSE).

**Exercise 4.13** *Repeat both forecasting methods from the example above but now with a training set of 10 years and the last two years as test set. Compute the RMSE for both methods.*

Disadvantages of using a standard prediction method such as linear regression are that trend changes are easily missed and that recent data has the same influence on the forecast as older data. For these reasons dedicated methods such as smoothing work better.

Smoothing in its simplest form works as follows. Suppose we have data without seasonality nor trend. Then two naive forecasts are: take the average of all data or just the last data point. The average filters out the noise in the best way, but uses data that is too old. Taking one data point has the disadvantage that it projects today's variability on the future. Smoothing is the perfect middle ground: it gives exponentially decreasing weights to the past. Let $y_1, \ldots, y_t$ be the observations up until now, time $t$. Then the forecast at $t$, $\hat{y}_t$, is given by:

$$\hat{y}_t = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots,$$

with $\alpha \in (0,1)$ some parameter. The nice thing about this form of smoothing (called *simple exponential smoothing* or SES) is that it gives a simple recursion:

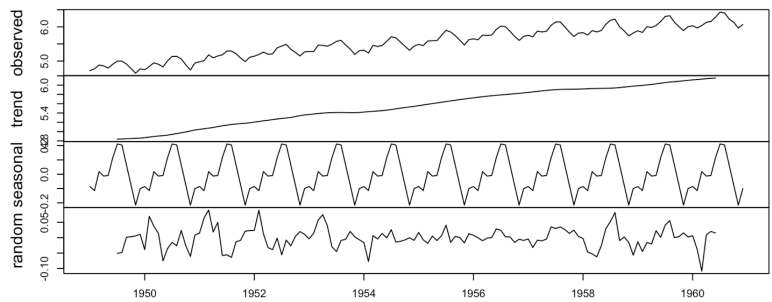
$$\hat{y}_t = \alpha y_t + (1-\alpha)\hat{y}_{t-1}. \tag{4.1}$$

This was especially handy in the days that computations still had to be done by hand.

SES can be extended to allow for slope, just as LR. This requires a slope forecast with a second smoothing parameter, usually called $\beta$. This method is known as *Holt's method*. When we also include smoothed seasonality factors then we get the *Holt-Winters method*. It is part of the `forecast` package and can be called with `hw()`.

**Exercise 4.14** *Apply Holt-Winters to the* `AirPassenger` *data and compute the RMSE in the same way as in Exercise 4.13.*

A method that we have not discussed yet is *decomposition*. Decomposition in its simplest form filters out the seasonality by taking *moving averages* of the length of the seasonal cycle. Then the remaining trend is estimated by, for example, linear regression. The average difference between trend and actual is the seasonal component. As an example, the R `prophet` forecasting library developed by Facebook Research for high-frequency data is based on decomposition. The `forecast` R library also has decomposition functionality. For the logs of the `AirPassenger` data we can use the command `plot(decompose(log(AirPassengers)))` leading to:




We see the original (log-transformed) data, the trend, the seasonal pattern, and the remaining errors.

**Exercise 4.15** *In this exercise we use the* `UKgas` *dataset. Apply the Holt-Winters method for different smoothing parameters. Apply ARIMA with a seasonal component using the* `auto.arima()` *function. Compare both methods using a plot and by computing the RMSEs on a well-chosen test set.*

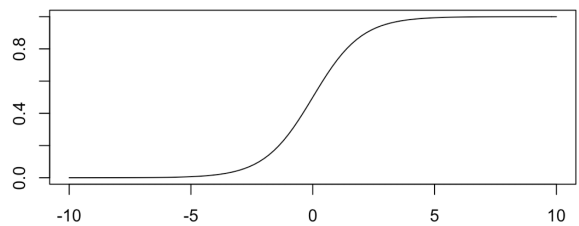
## 4.6 Classification

Up to now we considered numerical-valued target variables, which is generally called regression. In this section, we consider categorical target vari-

**Box 4.13. Explanatory variables**
Sometimes other variables offer additional information. They are called explanatory variables. In models based on linear regression they are easy to add. The ARIMA framework can be extended to ARIMAX to include these variables.

ables. This is called *classification*. We focus on the case of two classes; some methods can easily be extended to more than two. One by one we discuss how to extend the regression methods to classification.

Note that if we have two categories, classification amounts to selecting 0 or 1. Linear regression can be extended to classification by adding a function that maps the dependent variable to $[0, 1]$. The outcome is to be interpreted as the likelihood of 0 or 1, and a fit or prediction can be obtained by rounding. The function which is used is the *logistic function* $e^x/(1+e^x)$ (see below). For this reason this technique is called *logistic regression*.




After rounding to 0 or 1 the counts of the possible outcomes of the fit and the actuals can be tabulated in a $2 \times 2$ *confusion matrix*.

**Example 4.5** *We use the* `PimaIndiansDiabetes` *data from the* `mlbench` *library. We predict* `diabetes` *from the other attributes by*

```
> fit=glm(diabetes~.,data=PimaIndiansDiabetes,family=binomial)
> predict(fit,type="response")>0.5
```

*Here* `glm` *stands for* generalized linear model*, and* `family=binomial` *assures we are doing logistic regression. The predictions can by used together with a* `table()` *command to produce the confusion matrix:*

```
                actuals
predictions pos neg
        pos 156  55
        neg 112 445
```

*As an example, 112 people having diabetes were not classified as diabetic, so-called "false negatives". From the confusion matrix many different measures can*

*be derived, such as the fraction of rightly classified people, called the* accuracy*:* $(156+445)/(156+55+112+445) = 78\%$.

**Exercise 4.16** *Look up the definitions of sensitivity, specificity and precision (for example, at the Wikipedia entry of "confusion matrix") and compute them for the example above.*
*Compute sensitivity and specificity for different values of the cut-off point (which was 0.5) and put them both in a plot with the cut-off point ranging from 0 to 1.*

Deciding which measure to choose is not always easy. However, we should keep in mind that classification usually is not a goal by itself, but is used to make decisions. For example, regulations might require that the sensitivity is at least 95%, or it may be the case that false negatives are more expensive than false positives. This might help us determine the measure to use.

**Exercise 4.17** *Suppose in the example that false negatives are 10 times more expensive than false positives. What is the optimal cut-off point and what are the costs?*

**Box 4.14. Unbalanced datasets**
In the diabetes data 35% is positive. Therefore it is easy to obtain an accuracy of 65%: simply classify everybody as negative. This is an example of unbalanced data, and many datasets are even more unbalanced. Different methods exist to counter the averse consequences of unbalanced datasets, depending also on the technique used. They include changing the objective of the ML algorithm, and deleting or duplicating data to make the dataset balanced.

The `neuralnet` function can also be used to do classification with ANNs. To indicate the logistic activation function we use the following additional arguments: `act.fct="logistic", linear.output=FALSE`. The confusion matrix is as follows:

```
                actuals
predictions pos neg
        pos 190  77
        neg  78 423
```

As can be seen, the precision is a bit higher than for logistic regression. We used 2 hidden layers which contain each 2 nodes.

**Exercise 4.18** *For the* `PimaIndiansDiabetes` *dataset, run logistic regression and a neural net for training and test sets of 70 and 30% of the data. Which method do you prefer and why?*

It is very natural to use decision trees for classification. The command `rpart(diabetes∼.,data=PimaIndiansDiabetes)` gives the decision tree of Figure 4.1.

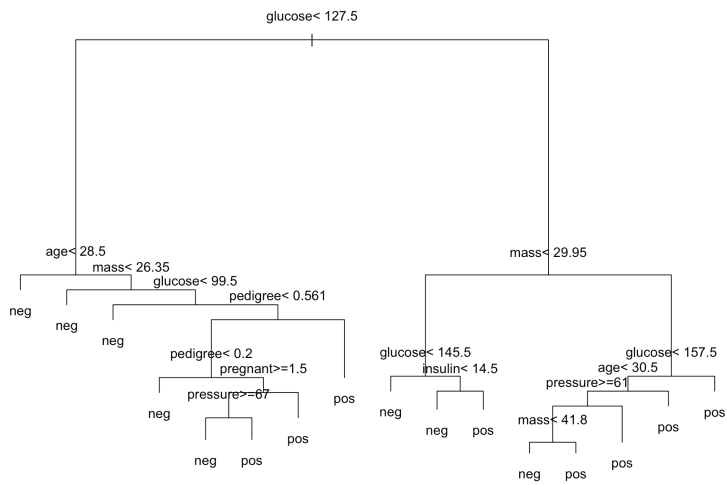



Figure 4.1: A decision tree for classification

**Exercise 4.19** *Determine the confusion matrix for the decision tree using the training and test set of Exercise 4.18.*

Random forests and gradient boosting can also be used for classification. We finish this section on classification with a technique that is specially designed for classification: *support-vector machines*. It creates the hyperplane (an $(n-1)$-dimensional plane in the $n$-dimensional feature space) that best separates the categories.

Calling `svm(diabetes∼.,data=PimaIndiansDiabetes)` from the `e1071` R package (named after a Vienna computer science department), leads to the following confusion matrix:

```
                actuals
predictions pos neg
        pos 170  37
        neg  98 463
```

**Exercise 4.20** *Determine the confusion matrix for the support-vector machine using the training and test sets of Exercise 4.18.*

## 4.7 Additional reading

`Kaggle.com` is an online platform for ML challenges where some of the world's best specialists compete for the prize money. Note that Netflix played an important role in popularizing competitions: the so-called *Netflix competition* consisted of building a recommender system for movies in the time that Netflix was still a relatively unknown company.

Good books on predictive modeling that are not overly technical are Kuhn & Johnson [23] and Hastie et al, [16]. Géron [12] also includes many details on how to get advanced neural networks working using TensorFlow, distributed computing, etc. The website `www.3blue1brown.com` contains some great videos on neural networks. Manning et al. [25] is a text book on information retrieval.

A good starting point for text mining in R is Silge & Robinson [38], which is also available online.

A practical and accessible text on forecasting is Hyndman & Athanasopoulos [18], which can also be read online. It makes extensive use of the R `forecast` library. More details on the M4 competition can be found in Makridakis et al. [24].

# Chapter 6

# Linear Optimization

In this chapter we discuss linear optimization problems. It is a framework used successfully in many industries to solve a broad variety of problems. We discuss different types of linear problems and show how you can solve them in Excel and R.

**Learning outcomes** On completion of this chapter, you will be able to:

- implement linear optimization problems in R, Excel and dedicated modeling languages
- model appropriate business problems as linear optimization models
- reflect on the usefulness and applicability of linear optimization

## 6.1 Problem formulation

We introduce linear optimization through an example. Later on we will give the general formulation.

Assume a company has $n$ products which it can produce using $m$ resources of which there is only a limited amount available. The problem to solve is which quantities to produce of each product such that the revenue is maximized and the resource constraints are satisfied. Examples of this *product-mix problem* are refineries combining different types of crude oil into end products, a farmer dividing his land between crops with constraints on the amount of fertilizer or environmental impact, etc.

Let us consider a simple *instance* of this problem. A company has 2 differents products, for example 2 types of crops, with profit 2 and 3 per quantity produced. We also have 2 resources: fertilizer and land. Product 1 requires 1 unit (e.g., ton) of fertilizer and 1 unit (e.g., acre) of land per unit produced, product 2 requires only 2 units of land. The availability of the resources is as follows: 5 units of fertilizer, 10 units of land. What is the optimal product mix?

This problem has a structure in which we can apply to many optimization problems:
- there are *decision variables* that have to be chosen;
- there is an *objective* that needs to be maximized;
- there are *constraints* which need to be satisfied.

In mathematical terms, our instance becomes:

$$\text{maximize } 2x_1 + 3x_2 \qquad \text{(objective)}$$

subject to

$$x_1 \leq 5 \qquad \text{(constraint resource 1)}$$
$$x_1 + 2x_2 \leq 10 \qquad \text{(constraint resource 2)}$$
$$x_1, x_2 \geq 0.$$

Because the functions $2x_1 + 3x_2$, $x_1$ and $x_1 + 2x_2$ are linear in $x = (x_1, x_2)$ we call this a *linear optimization* (LO) problem. For LO, efficient *solvers* exist that are guaranteed to give an optimal solution, even for problems with thousands of variables and constraints. To solve our instance with R, the following code can be used:

```
> install.packages("lpSolve")  (install solver package (use only once))
> library(lpSolve)                                        (load library)
> f.obj <- c(2, 3)                                        (set objective)
> f.con <- matrix (c(1, 0, 1, 2), nrow=2, byrow=TRUE)
                                                        (constraint values)
> f.dir <- c("<=", "<=")                                  (constraint types)
> f.rhs <- c(5, 10)                                      (resource amounts)
> lp ("max", f.obj, f.con, f.dir, f.rhs)                 (get optimal value)
> lp ("max", f.obj, f.con, f.dir, f.rhs)$solution
                                                      (get optimal solution)
```

**Exercise 6.1** *Solve the problem of Section 6.2 with R.*

There are different ways to understand what the R solver did. Let us first take a graphical look. In Figure 6.1 the problem is drawn with $x_1$ on

the horizontal axis and $x_2$ on the vertical one. We see the two constraints who together with the non-negativity constraints ($x_1, x_2 \geq 0$, assumed by default) delimit the allowable area, often called the *feasible region*. Because of the linearity of the constraints and the objective, the optimum (if it exists, see below) must be at the edge, at a corner. To determine the optimal corner, we slide a line with equal objective value until we hit the feasible region, i.e., the set of points that satisfy all constraints. The line with value 36 is drawn in the figure. When we slide it down it hits the feasible region in the point with $x_1 = 5$ and $x_1 + 2x_2 = 10$. From this, the optimal solution follows again: $x_1 = 5$ and $x_2 = 2.5$.

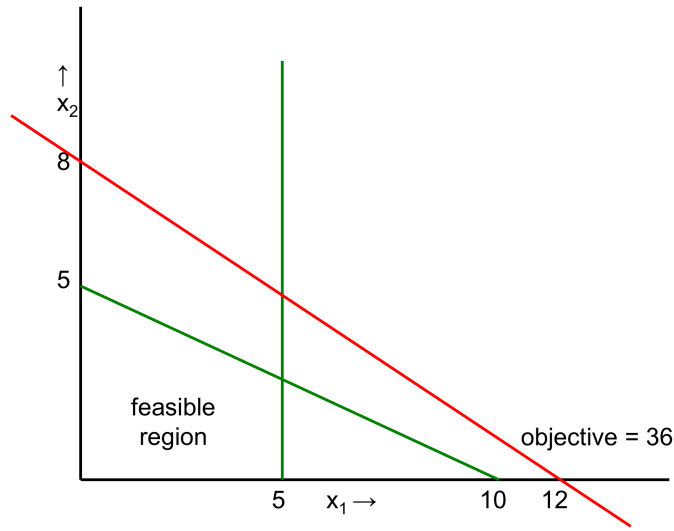



Figure 6.1: A graphical view of LO

Let us now take an algebraic point of view. The 2 constraints can be rewritten as equalities as follows, using additional variables $y_1$ and $y_2$:

$$\begin{array}{lcl} x_1 \leq 5 & & x_1 + y_1 = 5 \\ x_1 + 2x_2 \leq 10 & \Leftrightarrow & x_1 + 2x_2 + y_2 = 10 \\ x_1, x_2 \geq 0 & & x_1, x_2, y_1, y_2 \geq 0 \end{array}$$

We have 2 equalities and 4 variables. This means that 2 non-zero variables suffice to find a solution. All 4 corners of the feasible region correspond to such a solution. The interior corresponds to solutions for which all variables are positive. The corners correspond to the following solutions:

- $(x_1, x_2, y_1, y_2) = (0, 0, 5, 10)$, origin;
- $(x_1, x_2, y_1, y_2) = (0, 5, 5, 0)$, upper-left corner;
- $(x_1, x_2, y_1, y_2) = (5, 0, 0, 5)$, lower-right corner;
- $(x_1, x_2, y_1, y_2) = (5, 2.5, 0, 0)$, upper-right corner (the optimum).

The algorithm implemented in the solver hops from corner to corner until it cannot improve the objective value anymore. This method is called the *simplex algorithm*.

**Box 6.1. History of Linear Optimization**
Several researchers have formulated Linear Optimization problems but it was G.B. Dantzig (1914-2005) who invented in 1947 the simplex algorithm. Dantzig was an American scientist with German-French roots. At the time, until very recently, it was commonly known as linear *programming*. LO has been extremely useful for solving all kinds of business problems and is by far the most successful technique within operations research.
Its random, dynamic counterpart is called *dynamic programming* (see Chapter 9), developed by R.E. Bellman (1920-1984). This division between deterministic and random problems is still very visible in operations research theory and applications.

The general formulation of an LO problem is as follows, for $n$ decision variables and $m$ constraints:

$$\text{maximize } \sum_{i=1}^{n} p_i x_i \qquad \text{(objective)}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, \ldots, m \qquad \text{(constraints)}$$
$$x_1, \ldots, x_n \geq 0.$$

Instances where the objective needs to be minimized or with constraints of the form "$=$" or "$\geq$" can be rewritten to fit the general formulation.

**Exercise 6.2** *Consider the following LO problem:*

$$\textit{minimize } 2x_1 + x_2 + 4x_3 \qquad \textit{(objective)}$$

*subject to*

$$x_1 - 2x_2 + 2x_3 \leq 120$$
$$-x_1 - x_2 + 3x_3 = 100$$
$$x_1 - x_2 + x_3 \geq 80$$
$$x_i \geq 0 \textit{ for all } i.$$

*a. Solve it in R (see the online documentation of the R package* `lpSolve` *to find out how to change the signs of the constraints).*

*b. Rewrite it in the general form with maximization and "$\leq$" constraints.*
*c. Solve this problem in R.*

Sometimes we write the general formulation in matrix notation. Then it becomes:

$$\max\{p^T x | Ax \leq b, x \geq 0\},$$

where $p$ and $x$ are $n$-dimensional column vectors (and thus $p$ *transposed*, $p^T$, is a row vector), $b$ is an $m$-dimensional column vector, and $A$ is an $m \times n$ matrix.

Not all LO problems can be solved. Sometimes the problem is unbounded, meaning that solutions of arbitrarily large values can be found. On the other hand, there are also problems where there are no feasible solutions at all. In Figure 6.2 examples of both situations are shown.

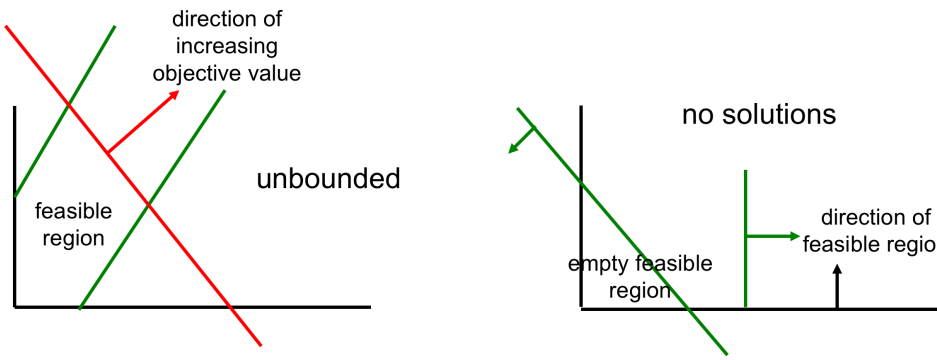



Figure 6.2: An unbounded (left) and an infeasible (right) LO problem

**Exercise 6.3** *Enter both problems of Figure 6.2 in R and see what output you get.*

## 6.2 LO in Excel

To solve LO problems in Excel, you should first make sure that the "solver add-in" is installed. You can check its availability under "Tools". How to install it depends on your version of Excel, for more details do a Google search for "install Excel solver".

Now we show how to solve the following problem using Excel:

$$\text{maximize } 2x_1 + 4x_2 + 8x_3 \qquad \text{(objective)}$$

subject to

$$x_1 + 3x_2 + 2x_3 \leq 10 \qquad \text{(constraint 1)}$$
$$x_1 + 3x_3 \leq 12 \qquad \text{(constraint 2)}$$
$$x_1, x_2, x_3 \geq 0.$$

The first step is to enter this problem in Excel in such a way that the decision variables, the objective value and the constraint values are in separate cells. See Figure 6.3 for a possible implementation of the above problem. For the decision variables we used arbitrary values $(1, 2, 3)$.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | Decision variables | | | | |
| 2 | | 1 | 2 | 3 | | |
| 3 | | Profit | | | objective | |
| 4 | | 2 | 4 | 8 | 34 | |
| 5 | Constraint | | | | | |
| 6 | number | matrix | | | values | limits |
| 7 | 1 | 1 | 3 | 2 | 13 | 10 |
| 8 | 2 | 1 | 0 | 3 | 10 | 12 |

Figure 6.3: LO implementation in Excel

The following formulas were entered:
- cell E4: `=SUMPRODUCT(B$2:D$2,B4:D4)`
- cell E7: `=SUMPRODUCT(B$2:D$2,B7:D7)`
- cell E8: `=SUMPRODUCT(B$2:D$2,B8:D8)`

Thanks to the $-signs, the formula only has to be entered once and can then be copied.

We are now ready to open the solver dialog. After entering the right values the dialog should look like Figure 6.4. The dialog can look slightly different, depending on your version of Excel. Hitting "Solve" will now solve the problem to optimality, with objective value 34.67.

**Exercise 6.4** *Solve the problem of Exercise 6.2 using Excel. Note that the constraints can be entered one by one each having a different sign.*

**Exercise 6.5** *The tax office can only check a subset of the tax declarations it received. There are 3 types of employees with different skills, and 3 types of declarations. Per declaration type, the expected revenues from additional taxation are as follows:* $(200, 1000, 500)$ *Euros. Every tax employee can process every declaration, except for employee type 2 who cannot process declaration type 2 and employee*

*type 3 who cannot process declaration type 3. The time per declaration depends on the declaration type and is* $(1,3,2)$ *hours, respectively, except for employee type 3 who takes 2 hours for a type 1 declaration. The numbers of declarations are* $(15000,6000,8000)$*, the numbers of available hours are* $(10000,20000,15000)$*. How do you assign the employees to the different declaration types? Use Excel to solve this problem.*

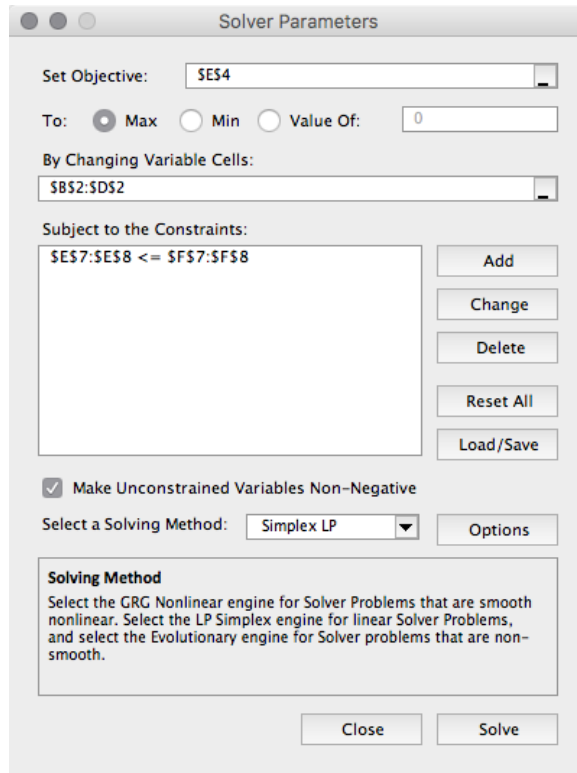



Figure 6.4: Excel solver dialog

The standard Excel solver is rather limited in capabilities. A simple alternative is the "OpenSolver" which is easy to install and comparable in use, but comes with a stronger solver without limitations on the numbers of variables or constraints. See `OpenSolver.org`.

**Exercise 6.6** *Extend the problem of Figure 6.3 in the following ways. When solving these problems it helps to think what the additional decision is that needs to be*

*taken.*
*a. Assume that, next to the 10 units available, you can buy extra units of resource 2 for the price of 1 per unit. What is the optimal solution now?*
*b. The same question, but now you can buy resource 1 for 2 per unit.*
*c. The same question, but now you can buy resource 1 for 1 per unit. Can you interpret the result?*

## 6.3 Example LO problems

In this section we consider several types of optimization problems that can be solved using LO.

A *graph* is the mathematical name for a network consisting of *nodes* and *edges* connecting the nodes. Nodes are sometimes also called *vertices*. Edges can be *directed* or *undirected*, i.e., uni-directional or bi-directional. Directed edges are often called *arcs*. Many practical problems can be formulated as problems on graphs. One such problem is *project planning*.

A project consists of a number of activities, each having a *duration*. These are the nodes in the graph. Additionally, certain activities require others to finish before they can get started. These precedence relations are modeled as directed edges in the graph. In Figure 6.5 an example of such a graph is given. We need to determine the earliest finish time of each activity. These are the decision variables, $x_i$ for activity $i$. Every precedence relation leads to a constraint. When $i$ precedes $j$ then this can be enforced by the constraint $x_i + d_j \leq x_j$, where $d_j$ is the duration of activity $j$. For example, in the project of Figure 6.5 we model the relation $F \to G$ by the constraint $x_F + 2 \leq x_G$.

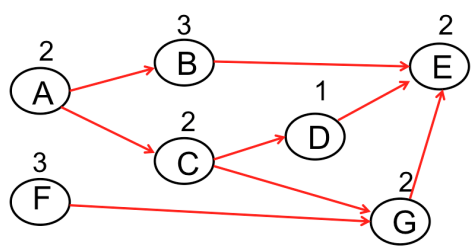



Figure 6.5: Directed graph with weights of a project planning problem

We are interested in the time at which all activities are finished. This can be modeled by an additional variable $z$, bigger than all finish times, that has to be minimized. This leads to the following LO formulation:

minimize $z$
subject to
$z \geq x_i$ for all vertices $i$
$x_i + d_j \leq x_j$ if $i$ precedes $j$
$x_i \geq d_i$ for all vertices $i$.

The last constraint ensures that no activity starts before time 0. Note that the finish time can also be found using an algorithm without LO, and that this algorithm can be extended to random activity durations. This is relevant in practice because often durations are hard to predict accurately, which is one of the main reasons why, for example, IT projects often finish after the scheduled deadline.

**Exercise 6.7** *Formulate the project planning problem of Figure 6.5 as LO problem and solve it using Excel.*

A more complicated problem that can be solved using LO is the so-called *transportation problem*. In this problem we have to transport a single type of good from $n$ sources to $m$ destinations. Source $i$ has supply $a_i$, destination $j$ has demand $b_j$, and link $i \to j$ has transportation costs $c_{ij}$ per unit transported on it. The question is: For each link, how much should you ship on it in order to satisfy the demand of each destination without violating supply constraints? See Figure 6.6 for an illustration. When link $i \to j$ does not exist we can take $c_{ij} = \infty$.

The LO formulation is as follows:

$$\text{minimize } \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

subject to

$$\sum_{j=1}^{m} x_{ij} \leq a_i \text{ for } i = 1, \ldots, n;$$

$$\sum_{i=1}^{n} x_{ij} \geq b_j \text{ for } j = 1, \ldots, m;$$

$$x_{ij} \geq 0 \text{ for all } i, j.$$

**Exercise 6.8** *Solve the following transportation problem, where "x" means there is no connection.*

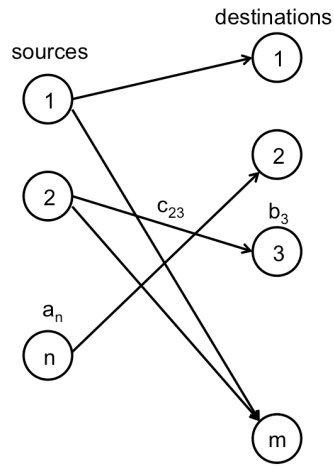



Figure 6.6: Transportation problem

| | | $j=1$ | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $i$ | $a_i$ | | $c_{ij}$ | | |
| 1 | 10 | 0 | 5 | $x$ | 0 |
| 2 | 3 | 6 | 4 | 6 | 4 |
| 3 | 6 | 2 | 10 | 2 | 4 |
| 4 | 6 | 6 | $x$ | 4 | 6 |
| | $b_j$ | 5 | 5 | 5 | 5 |

If we add intermediate nodes to the transportation problem, as in Figure 6.7, then we obtain the *transshipment problem*. We can solve it by adding constraints of the form:

$$\sum_{i=1}^{n} x_{ik} = \sum_{j=1}^{m} x_{kj}$$

for all intermediate nodes $k$. It can be extended to a network by adding multiple layers of intermediate nodes.

Next we consider *multi-period production/inventory models*. Here we assume there are multiple time periods, say $t = 1, \ldots, T$, and a starting inventory $s_0$. Every day, the amount of production or supply has to be decided: $x_t$. There is demand, $d_t$ at time $t$, holding costs $h_t$, and production costs $c_t$. The problem is to find the production/order schedule that minimizes the total costs. This can be formulated as an LO problem as follows:

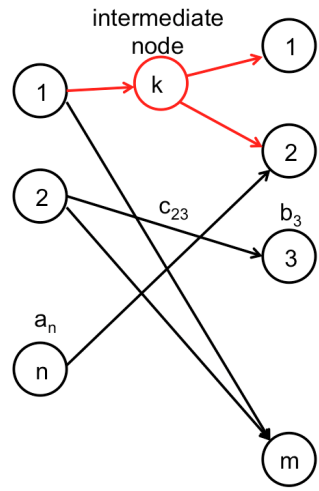



Figure 6.7: Transshipment problem

$$\text{minimize} \sum_{t=1}^{T} (c_t x_t + h_t s_t)$$

subject to

$$s_{t+1} = s_t - d_{t+1} + x_{t+1} \text{ for } t = 1, \ldots, T-1;$$
$$x_t, s_t \geq 0 \text{ for } t = 1, \ldots, T.$$

This model can be extended in many different directions, such as maximum stock or production capacity, multiple products and resources, backorders, and fixed order costs (see Exercise 6.19).

## 6.4 Integer problems

For the simplex algorithm to be used, it is essential that the objective and all constraints are linear. Many extensions exist to non-linear functions. One important class is where there is, in addition to the linear constraints, constraints requiring one or more of the decision variables to be integer (i.e., taking values in $\{0, 1, 2, \ldots\}$) or binary (taking values in $\{0, 1\}$). We call these *integer linear optimization* (ILO) problems.

Note that binary problems are special cases of integer problems: the constraint $x_i \in \{0, 1\}$ is equivalent to the following 2 constraints: $x_i \in \{0, 1, 2, \ldots\}$ and $x_i \leq 1$.

Product-mix problems where we have to produce integer numbers of items is a good example. In R we can add an additional argument to the solver call:

```
> lp ("max", f.obj, f.con, f.dir, f.rhs, all.int=TRUE)
```

In Excel we have to add additional constraints, as in Figure 6.8.

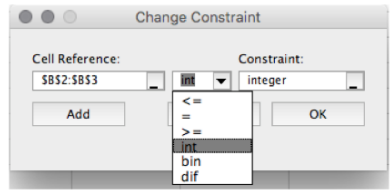



Figure 6.8: Entering integer and binary constraints in Excel

**Exercise 6.9** *Solve the integer version of the problem of Section 6.1.*

The archetypical binary LO problem is the *knapsack problem*. You have to make a selection out of a set of items. Each item has a revenue and a weight. The goal is to maximize the total revenue with a constraint on the total weight. Typical applications of the knapsack are logistics problems, for example selecting items which have to be transported in trucks, or so-called cutting problems, which arise, for example, in steel plants where you have to cuts plates in pieces of different sizes.

As an example, consider a problem with total weight capacity 11. The items are as follows:

| revenue | 60 | 60 | 40 | 10 | 20 | 10 | 3 |
|---|---|---|---|---|---|---|---|
| weight | 3 | 5 | 4 | 1.4 | 3 | 3 | 1 |

Formulated as ILO we get:

maximize $60x_1 + 60x_2 + 40x_3 + 10x_4 + 20x_5 + 10x_6 + 3x_7$

subject to

$3x_1 + 5x_2 + 4x_3 + 1.4x_4 + 3x_5 + 3x_6 + x_7 \leq 11$

$x_i \in \{0, 1\}$ for all $i$.

Solving this using R or Excel leads to the optimum $(1, 1, 0, 0, 1, 0, 0)$ with value 140.

**Exercise 6.10** *Verify that this is indeed the optimal solution by solving the problem in R and Excel.*

Although the solver seemed to have found the optimal answer without any problems, it required much more work. This becomes apparent when we solve big real-life ILO problems with hundreds or thousands of variables. To gain more insight in how ILOs are solved, let us have a look at Figure 6.9, where we see the steps to solve the knapsack example. We start with solving the LO *relaxation*, which is the problem without the integer or binary constraints (step 1). Sometimes we find an integer solution right away. Certain types of problems are even guaranteed to give integer solutions immediately. Here however $x_3$ is non-integer. Its value (150) is an upper bound to the best integer solution.

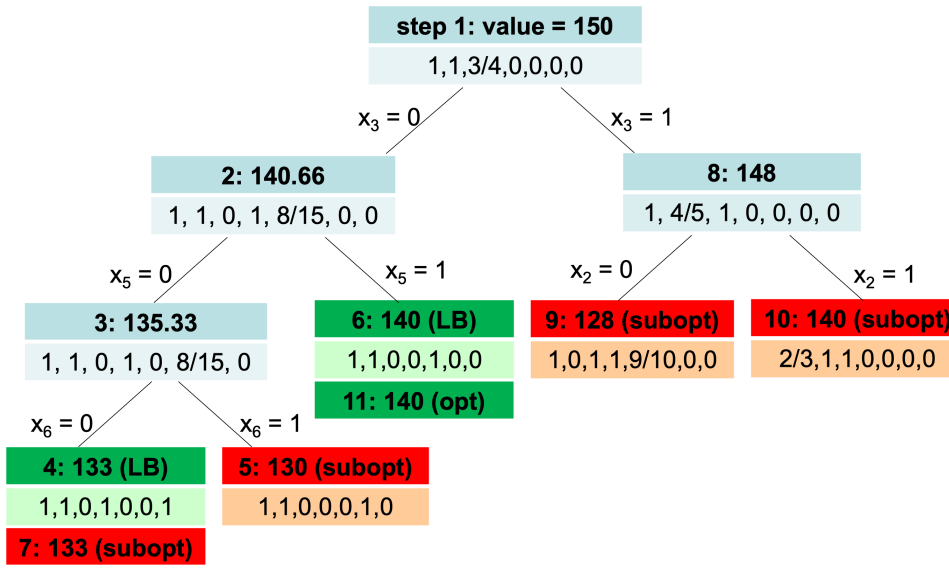



Figure 6.9: Solving an ILO problem

Now we *branch* on $x_3$, and we continue with the branch $x_3 = 0$. We solve the relaxation again, but with $x_3 = 0$. We find again a non-integer solution (step 2). We continue branching until we find an integer solution in step 4 with value 133. It is called a *lower bound* (LB) of the optimum: perhaps there are other integer solutions with values between 133 and 150. To find out if there are any such solutions we work our way back up to make sure all branches are dealt with. In step 5, we find an integer solution that is worse than the LB. In step 6, we find a higher binary value than the LB. It becomes the new LB, and the old LB is now sub-optimal (step 7). We have dealt with

the left side of the tree, we move to the right. In step 8, we find a non-integer solution, we branch on $x_2$. In step 9 and 10, we find non-integer solutions which are worse or equal than the LB. Adding constraints will not make the value higher, therefore these branches can be discarded. We have dealt with all branches, and therefore the current LB is the optimum (step 11). This algorithm is called *branch-and-bound*.

Many LO solvers can also handle integer constraints. However, not all solvers can solve big instances. The best solvers are proprietary, notably CPLEX and Gurobi.

**Exercise 6.11** *Solve by branch-and-bound the knapsack problem having rewards* $(15, 9, 10, 5)$*, sizes* $(1, 3, 5, 4)$ *and capacity 8. Check the result with R.*

## 6.5 Example ILO problems

In this section we discuss a number of problems that can be solved with ILO. The first is the *set cover problem*. We have a so-called *universe* $U = \{1, , \ldots, m\}$, and sets $S_1, \ldots, S_n$, with $S_i \subset U$. We are looking for the smallest selection of sets that covers $U$.

As an example, let $U$ be set of locations where incidents can happen, and every set $S_i$ the set of locations that can be reached by an ambulance from a certain base station within a certain target time. Then the set cover problem is as follows: what is the minimum number of ambulances needed and what are their base stations such that all locations can be reached within the target time?

The ILO formulation is as follows:

$$\text{minimize} \sum_{i=1}^{n} x_i$$

subject to

$$\sum_{i:u\in S_i} x_i \geq 1 \text{ for all } u \in U;$$

$$x_i \in \{0, 1\} \text{ for all } i.$$

The binary constraints are necessary, as the following example shows. Let $U = \{1, 2, 3\}$ and $S_1 = \{1, 2\}$, $S_2 = \{1, 3\}$, and $S_3 = \{2, 3\}$. Then any combination of 2 sets is optimal but the LO relaxation has optimal value 1.5 with solution $(0.5, 0.5, 0.5)$.

The main constraint is often replaced by the following more convenient notation: $\sum_{j=1}^{n} a_{uj} x_i \geq 1$ with $a_{uj} = 1$ if $u \in S_j$, 0 otherwise.

**Exercise 6.12** *Solve the following ILO problem, inspired by [13]. A swimming pool is open during 12 hours, and the lifeguards at duty should be selected. Every lifeguard has his/her own working hours and wage, as given in the table. Select the optimal combination of lifeguards assuring at least 1 lifeguard at every moment. The hours mentioned are the first and last hour that each lifeguards works, thus Ben/Celia/Fred is a feasible solution.*

| *Lifeguard* | *Ann* | *Ben* | *Celia* | *Dick* | *Estelle* | *Fred* |
|---|---|---|---|---|---|---|
| *Hours* | *1–6* | *1–4* | *5–8* | *7–10* | *7–12* | *9–12* |
| *Wages* | *8* | *6* | *6* | *3* | *7* | *3* |

*Solve it in Excel and use the* `SUMPRODUCT` *function and a matrix with the values of* $a_{ui}$.

A generalization of the set cover problem is the *covering problem*. Instead of having to cover each element of the universe by 1 it can be more general. This leads to the following problem formulation:

$$\text{minimize } \sum_{i=1}^{n} x_i$$

subject to

$$\sum_{i=1}^{n} a_{ui} x_i \geq b_u \text{ for all } u \in U;$$
$$x_i \in \mathbb{N}_0 = \{0, 1, 2, \ldots\} \text{ for all } i.$$

This problem can be applied to *shift scheduling*, a problem already introduced by Dantzig in [7]. In shift scheduling, we have to find the best combination of shifts of employees, which in total cover the required workforce in every time interval. The seminal problem studied by Dantzig involved employees at a toll station, where the required coverage fluctuates during the day.

To model this as a covering problem, let $U$ be the set of time intervals. Every set $S_i$ corresponds to a shift (with $a_{ui} = 1$ meaning shift $i$ works at time $u$), and $b_u$ the number of required workers at time $u$. Then $x_i$ corresponds to the number of employees that need to have shift $i$. By adding a coefficient to $x_i$ in the objective we can add different costs to the shifts.

**Exercise 6.13** *The required staffing in a call center, from 9am to 9pm in 30-minute intervals, is as follows:*
*10, 11, 13, 16, 16, 13, 11, 10, 10, 11, 12, 13, 14, 14, 13, 11, 10, 9, 9, 10, 9, 8, 8, 8.*
*There are 2 types of shifts:*
*- 8 hours working time, with a 30-minute unpaid break in the middle, wage 20*

*Euro/hr, possible starting times every half hour from 9am to 12:30pm;*
*- 4 hours consecutive, wage 24 Euro/hr, possible starting times every half hour from 9am to 5pm.*

*Formulate this as a covering problem and solve it with the Excel solver.*

*Machine scheduling* is another important class of ILO problems. However, because it involves some modeling tricks that are discussed in Section 6.7, we defer discussing it to that section.

The next two exercises concern problems that can be solved with appropriately chosen binary decision variables.

**Exercise 6.14** *For a day at a school, classes have to be assigned to professors such that each class has an hour with each required professor and such that there are no conflicts such as a professor having to teach two classes at the same time. A matrix with entries $a(c,p)$ indicates which classes need to have which professors: when $a(c,p)=1$ then class $c$ needs to have professor $p$, otherwise $a(c,p)=0$.*

*Formulate an ILO model that minimizes the total number of hours that classes have to spend at school. A class remains at school until right after the last hour it has seen a professor.*

**Exercise 6.15** *For a classroom assignment, pairs need to be made of $n$ students. Each student can give a list of students he or she is willing to work with.*

*Formulate an ILO model that maximizes the number of pairs that can be made. Each student is only allowed to be part of one pair, but it might not be possible to assign all students to a pair.*

## 6.6 Modeling tools

So far we discussed solving LO and ILO problems, and the engines to solve them. However, we should realize that optimization specialists spend most of their time on *modeling*. Modeling is the translation of a real-life problem into a mathematical description that can be used to solve the problem. See Figure 6.10 for the steps in modeling. The time spent on modeling can be greatly reduced by using an appropriate modeling tool or language. As such, the existence of these modeling tools is considered to be of equal importance as the engines used to solve the models. To learn a modeling tool or language requires some time, but this easily pays off if you often build models for optimization problems.

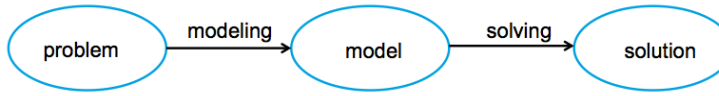



Figure 6.10: Modeling steps

The best known *algebraic modeling languages* (AMLs) are AIMMS, AMPL, GAMS, LINDO, and MPL. Some of these languages are part of an *integrated development environment* (IDE) that simplifies the modeling even further. The problem entry in these AMLs is quite similar to mathematical notation (see Box 6.2 on AMPL). Many of the tools and engines have free educational licenses, which makes it possible for students to learn and experiment. Even simpler to use is the *NEOS server*, a free cloud service which features a range of solvers to which one can submit optimization problems in a number of AML formats.

The AMLs and associated IDEs allow experienced modelers to model and solve problems they encounter. However, engines can also be built into software dedicated to solve a particular class of problems such as navigation software. These types of problem-specific tools are called *decision support systems* (DSSs). They are geared towards a different class of users. While AMLs are used by experienced data scientists and OR consultants, DSSs are most often used by planners with domain knowledge, but with less or no background in optimization and modeling.

## 6.7 Modeling tricks

In the previous section we saw which tooling to use. In this section we will see how to put problems in the (I)LO format. Thinking in the decision variables-objective-constraints framework often allows you to arrive at a problem formulation that resembles the standard (I)LO formulation. However, sometimes it is difficult to ensure that objective and constraints are linear. This section discusses some often-used tricks to formulate certain types of objectives and constraints in a linear way.

The first trick is useful when we want to minimize the absolute value of some decision variable, thus when the problem is of the form

$$\min\{c^T|x||Ax \leq b\},$$

for some vector $c \geq 0$.

**Box 6.2. A knapsack problem in AMPL**

We give the AMPL implementation of the knapsack problem and a small instance which can be submitted right away to the NEOS server. The problem structure is implemented in the model file:

```
set ITEMS;

param size {ITEMS};
param total_size;
param revenue {ITEMS};

var take {ITEMS} binary;

maximize Total_Revenue:
        sum {i in ITEMS} take[i] * revenue[i];

subject to Size_Constraint:
        sum {i in ITEMS} take[i] * size[i] <= total_size;
```

Next we need a data file in which the instance is given and finally the run file which tells the NEOS server what to do:

```
data;

set ITEMS := 1 2 3 4 5;

param: size :=
1  10
2  8
3  6
4  4
5  2;

param total_size := 15;

param: revenue :=
1  10
2  4
3  4
4  4
5  1;
```

```
solve;
display take;
```

Note that the problem structure and data are separated. When a planner has to solve a knapsack problem every day, he only needs to change the data file. Further details on the AMPL syntax can be found online or in the AMPL book [10].

The crucial idea is that the variable $x_i$ can be rewritten as follows: $x_i = x_i^+ - x_i^-$ with $x_i^+, x_i^- \geq 0$ and one of them 0. Now the optimization problem can be rewritten as follows:

$$\min\{c^T(x^+ + x^-) | A(x^+ - x^-) \leq b, x^+, x^- \geq 0\},$$

which is linear. Because $c \geq 0$ for each $i$ either $x_i^+ = 0$ or $x_i^- = 0$.

**Exercise 6.16** *Numbers $a_1, \ldots, a_n$ are given. We are looking for $x$ that minimizes $\sum_i |x - a_i|$. Formulate this as LO problem, and implement it in Excel for the following numbers: 1, 2, 3, 5, 8, 10, 20, 35, 100. How can you interpret the outcome?*

The previous exercise shows that the median minimizes the sum of absolute errors, much as the average minimizes the sum of squared errors. We

can extend this to linear functions. We already did this for squared errors, for which *linear regression* is the method. For absolute errors it is called *quantile regression*. Points $(x_i, y_i)$ are given and the objective is to find a function $y = a + bx$ such that the sum of absolute errors is minimized, see Figure 6.11.

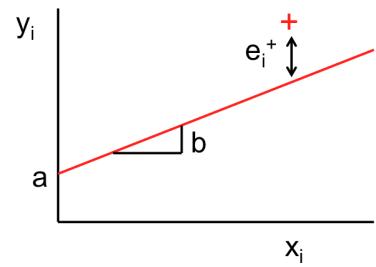



Figure 6.11: Quantile regression

In vector notation the problem can be formulated as follows:

$$\min\{\mathbf{1}^T|e| \mid y - (a\mathbf{1} + bx) = e\},$$

with $\mathbf{1}$ a vector with only 1's, and $e_i = y_i - (a + bx_i)$ the errors as given in the constraint. This can be made linear as follows:

$$\min\{\mathbf{1}^T(e^+ + e^-) \mid y - (a\mathbf{1} + bx) = e^+ - e^-, \quad e^+, e^- \geq 0\}.$$

We can generalize this to an asymmetric objective in the following way:

$$\min\{p\mathbf{1}^T e^+ + (1-p)\mathbf{1}^T e^-) \mid y - (a\mathbf{1} + bx) = e^+ - e^-, \quad e^+, e^- \geq 0\},$$

with $0 < p < 1$. This explains the term quantile regression.

**Exercise 6.17** *Consider Exercise 6.13. Assume we only have 8-hour shifts. To avoid overstaffing we replace the condition that staffing is met in every interval by the following objective: minimize the sum of absolute differences between demand and schedule. Formulate this as a LO problem and solve it using Excel.*

The next modeling trick is for cases where the objective function is not a sum but a maximum. The full problem is then of the form

$$\min\{\max\{x_1, \ldots, x_n\} \mid Ax \leq b,\ x \geq 0\}.$$

When discussing project planning we already say how this can be put in the LO framework:

$$\min\{z \mid z\mathbf{1} \geq x,\ Ax \leq b,\ x \geq 0\}.$$

Finally, there are some uses of a very big number, often called *big M*, in the context of ILO. The first is when an *indicator function* is part of the objective. An indicator function is a 0/1 function which is equal to 1 when a condition is satisfied. As an example, take the transportation problem with fixed costs $K$ when a link is used. To model this, we introduce binary variables $y_{ij}$ such that $y_{ij} = 1 \Leftrightarrow x_{ij} > 0$. Now we can simply add $K \sum_{i,j} y_{ij}$ to the objective function. But how to set $y_{ij}$? Here the value $M$ comes into play, by adding the following constraints:

$$My_{ij} \geq x_{ij}.$$

We assume $M$ is bigger than any $x_{ij}$ ever can be. Thus, $x_{ij} > 0 \Rightarrow My_{ij} > 0 \Rightarrow y_{ij} = 1$. When $x_{ij} = 0$ then $y_{ij}$ can be 0 or 1. Because we are minimizing costs and $K > 0$, $y_{ij}$ will be 0. Thus $y_{ij} = 1 \Leftrightarrow x_{ij} > 0$.

**Exercise 6.18** *Consider Exercise 6.8, but with an additional feature: every link that is used has fixed costs 5. Determine the optimal solution, using ILO.*

**Exercise 6.19** *Consider the multi-period production/inventory model of page 94. Extend it to fixed order costs, meaning that costs K are incurred at t when $x_t > 0$, keeping all constraints linear.*

Big $M$ can also be used in other situations, notably when a constraint only has to hold when a condition, which is part of the decision variables is satisfied. This condition is represented by a binary variable, let's say $y$, and the constraint is of the form $x \leq b$. Then a linear implementation is $x \leq b + (1-y)M$. When $y = 0$ the contraint always holds because the right-hand side is very big. When $y = 1$ then we find the original $x \leq b$.

As an example, consider 2 jobs, $A$ and $B$, that have to be executed consecutively, but the order is a decision to be made. Let $x_A$ ($x_B$) be the starting time of job $A$ ($B$), and $d_A$ ($d_B$) the duration of $A$ ($B$). When $A$ goes before $B$ then we get the condition $x_A + d_A \leq x_B$, otherwise $x_B + d_B \leq x_A$. This can be modeled in a linear way by having the following conditions:

$$x_A + d_A \leq x_B + (1-y)M, \qquad x_B + d_B \leq x_A + yM, \qquad y \in \{0,1\}.$$

Here $y = 1$ corresponds to $A$ before $B$.

**Exercise 6.20** *Assume that activities B and C of the project planning problem of Figure 6.5 use the same resource and therefore cannot be scheduled at the same time. Formulate this as ILO problem and solve it using Excel. The shortest finish time is 9.*

**Machine scheduling** A problem in which several of these concepts occur is *machine scheduling*. We consider jobs that need to be scheduled on a single machine. Job $i$ has release date $r_i$ before which it cannot start, duration $d_i$, and due date $t_i$, $i = 1, \ldots, n$. The decision variable are $x_i$, when to start job $i$, and also binary variables $y_{ij}$ with $y_{ij} = 1$ iff (read: if and only if) job $i$ goes before $j$, for all $i \neq j$. We give all constraints and discuss objectives right after that:

$$x_i + d_i \leq x_j + My_{ji} \text{ for all } i, j \text{ such that } i \neq j, M \gg 0 \qquad \text{(overlap)}$$
$$y_{ij} + y_{ji} = 1 \text{ for all } i, j \text{ such that } i \neq j \qquad \text{(order)}$$
$$x_i \geq r_i \text{ for all } i \qquad \text{(release dates)}$$
$$y_{ij} \in \{0, 1\} \text{ for all } i, j \text{ such that } i \neq j$$

Different objectives are possible. Some common ones are:
- the *flowtime*, defined as $\sum_i (x_i + d_i - r_i)$, thus the sum of the times that the job is waiting to be processed or is being processed; that is, the time they spend "in the system";
- the *makespan*, defined as $\max_i \{x_i + d_i\}$, the time when the machine is ready with all jobs;
- the total *tardiness*, which is $\sum_i (x_i + d_i - t_i)^+$, the sum of the times that the jobs are late, counting finishing early as 0.

The makespan can be modeled by an additional decision variable $z$ that needs to be minimized and which needs to satisfy: $x_i + d_i \leq z$ for all $i$.

The total tardiness can be modeled using additional decision variables $z_i$ representing the tardiness of job $i$. The objective becomes $\min \sum_i z_i$, and we get additional constraints $x_i + d_i - t_i \leq z_i$ and $z_i \geq 0$ for all $i$.

The model quickly becomes big, also for moderate $n$: the number of variables is $n^2$, and the number of constraints is $2n^2 + n$.

**Exercise 6.21** *Implement the single-machine scheduling problem with tardiness as objective in AMPL. Solve it for the following data with an appropriate solver on the NEOS server:*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *duration* | 4 | 5 | 3 | 5 | 7 | 1 | 0 | 3 | 2 | 10 |
| *release time* | 3 | 4 | 7 | 11 | 10 | 0 | 0 | 10 | 0 | 15 |
| *due date* | 11 | 12 | 20 | 25 | 20 | 10 | 30 | 30 | 10 | 20 |

*To implement a constraint that needs to hold for all $i, j$ with $i \neq j$ you can use the following AMPL syntax:*

```
subject to example_constraint {i in 1..N, j in 1..M: i<>j}:
```

*2-dimensional binary variables are defined as follows:*

```
var x {1..N, 1..M} binary;
```

**Exercise 6.22** *Change the objective of Exercise 6.14 as follows: the time the last class finishes has to be minimized.*

## 6.8 Additional reading

Books on optimization come in two flavors: either they promote optimization to executives without going into any technical details (such as [35]), or they are written for students with a strong theoretical background such as industrial engineering. Books having a considerable overlap with the current chapters and containing many examples of optimization problems are introductions to MS/OR such as [17, 40, 42] and Rardin [32] which has a more narrow focus on deterministic optimization. Foreman [9] is another accessible book with some chapters on optimization (especially the last part of Chapter 1 and Chapter 4).

Project management is extremely important in practice and has many different scientific aspects. Project planning is one of these aspects which we only basically touched upon. A good starting point for further reading is Klastorin [21]. See also Section 5.1.

More details on the AMPL modeling language can be found in [10]. See `en.wikipedia.org/wiki/List_of_optimization_software` for a list of solvers. The NEOS server can be reached at `neos-server.org/neos/`. The AIMMS optimization modeling book [3] gives many modeling tricks.

In the presence of excellent LO and ILO solvers, much research effort is currently put into non-linear optimization. A a result, much progress has been made in this area over the last decade.

# Bibliography


[1] INFORMS Analytics Maturity Model. Available online at `https://analyticsmaturity.informs.org`.

[2] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[3] J. Bisschop. *AIMMS Optimization Modeling*. AIMMS, Haarlem, 2016. Downloadable from AIMMS website.

[4] G. Chaslot, M. Winands, van den Herik, H., J. Uiterwijk, and B. Bouzy. Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, 4:343–357, 2008.

[5] V. François-Lavet, P. Henderson, R. Islam, M.G. Bellemare, and J. Pineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3–4), 2018.

[6] R.G. Cross. *Revenue Management: Hard-Core Tactics for Market Domination*. Broadway Books, 1998.

[7] G.B. Dantzig. A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[8] T.H. Davenport and J.G. Harris. *Competing on Analytics: The New Science of Winning*. Harvard Business School, 2007.

[9] J.W. Foreman. *Data Smart*. Wiley, 2013.

[10] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury, Thomson, 2003. Available online at `http://ampl.com/resources/the-ampl-book`.

[11] M.C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14:192–215, 2002.

[12] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly, 2017.

[13] B. Guenin, J. Könemann, and L. Tunçel. *A Gentle Introduction to Optimization*. Cambridge University Press, 2014.

[14] C.M. Harris, S.P. Murphy, and M. Vaisman. *Analyzing the Analyzers*. O'Reilly Media, 2013.

[15] P. E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2017.

[17] F.S. Hillier and G.J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 8th edition, 2005.

[18] R.J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. O Texts, 2018.

[19] L.C.M. Kallenberg. Markov Decision Processes. Unpublished textbook. Downloadable from `http://goo.gl/5nnKN7`, 2018.

[20] W.D. Kelton, R.P. Sandowski, and D.A. Sandowski. *Simulation with Arena*. McGraw-Hill, 1998.

[21] T. Klastorin. *Project Management: Techniques and Tradeoffs*. Wiley, 2003.

[22] J.P. Kotter. *Leading Change*. Harvard Business School Press, 1996.

[23] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer, 2013.

[24] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34:802–808, 2018.

[25] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[26] B.L. Nelson. *Foundations and Methods of Stochastic Simulation*. Springer, 2013.

[27] K. O'Neil. *Weapons of Math Destruction*. Crown Books, 2016.

[28] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[29] S.G. Powell, K.R. Baker, and B. Lawson. Impact of errors in operational spreadsheets. *Decision Support Systems*, 7:126–132, 2009.

[30] W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2nd edition, 2011.

[31] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[32] R.L. Rardin. *Optimization in Operations Research*. Pearson, 2014.

[33] S.M. Ross. *Simulation*. Academic Press, 6th edition, 1996.

[34] S.M. Ross. *A First Course in Probability*. Prentice Hall, 6th edition, 2002.

[35] S. Sashihara. *The Optimization Edge*. McGraw Hill, 2011.

[36] S. Savage. *The Flaw of Averages: Why We Underestimate Risk in the Face of Uncertainty*. Wiley, 2012.

[37] D. Schultes. *Route Planning in Road Networks*. PhD thesis, Universität Fridericiana zu Karlsruhe, 2008.

[38] J. Silge and D. Robinson. *Text Mining with R*. O'Reilly, 2018.

[39] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2000.

[40] H.A. Taha. *Operation Research: An Introduction*. Prentice Hall, 6th edition, 1997.

[41] M.F. Triola. *Elementary Statistics*. Pearson, 13th edition, 2017.

[42] W.L. Winston. *Operations Research: Applications and Algorithms*. Duxbury Press, 1987.

# Index