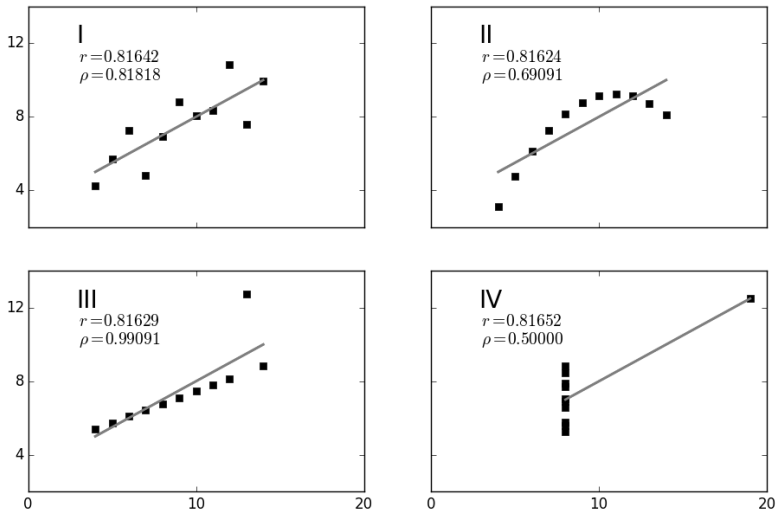
MSc in Life Sciences
Core Competences – Handling and Understanding Data

D1

Handling and Visualising Data



Dr. Manuel Gil
Dr. Simone Ulzega

Contents

1 Introduction **5**

2 Databases **7**
2.1 Principles of Relational Databases (RDBs) 10
2.1.1 Structure of RDBs 10
2.1.2 Querying RDBs 17
2.2 Introduction to the Semantic Web 20
2.2.1 RDF 22
2.2.2 Merging data 22
2.2.3 RDF Schema and URIs 23
2.2.4 Ontologies, OWL, Inference 24
2.2.5 OBO 25
2.2.6 RDF and RDBs 26
2.2.7 Real example of data integration 28

3 Data preparation in R **31**
3.1 Data Tidying 31
3.1.1 Prerequisites 31
3.1.2 Data frames of type *tibble* 32
3.1.3 Structure of datasets 33
3.1.4 Meaning of datasets 34
3.1.5 Tidy data 34
3.1.6 Tidy works well with R 35
3.1.7 From messy to tidy 35
3.1.8 Concluding remarks 37
3.2 Missing Data 38
3.2.1 Classifying missing data 38
3.2.2 Handling missing data 39

4 Data visualisation **43**
4.1 Introduction to ggplot 43
4.1.1 The underlying grammar of ggplot 44
4.1.2 ggplot by examples with basic graphs 46
4.1.3 Aesthetics and more than one layer 50

4.1.4 Faceting, multiple plots, saving plots . . . 51
4.1.5 And now? . . . 52
4.2 Special plots: Tufte in R with ggplot . . . 52

5 Exploring and Describing Data 53
5.1 Introduction . . . 53
5.1.1 Statistics in a nutshell . . . 53
5.1.2 Population and sample . . . 54
5.1.3 Data . . . 56
5.1.4 Sampling . . . 60
5.1.5 Frequency distributions . . . 63
5.2 Organizing and graphing data . . . 67
5.2.1 Raw data . . . 67
5.2.2 Analysis of qualitative data . . . 68
5.2.3 Analysis of quantitative data . . . 72
5.2.4 Graphical representation of quantitative data . . . 75
5.2.5 Cumulative frequency distributions . . . 80
5.2.6 More tools for quantitative data (I). Stem-and-leaf displays. 83
5.2.7 More tools for quantitative data (II). Dot plots. . . . 84
5.3 Numerical descriptive measures . . . 86
5.3.1 Measures of central tendency . . . 86
5.3.2 Measures of dispersion . . . 98
5.3.3 Descriptive measures for grouped data . . . 103
5.3.4 Meaning of the standard deviation . . . 104
5.3.5 Measures of position . . . 106
5.3.6 Box-and-Whisker plot . . . 111

6 Project Work 115
6.1 Importing, First Inspection, and Cleaning . . . 115
6.2 Exploratory Analysis: Overview . . . 116
6.3 Exploratory Analysis: Detailed View . . . 116
6.4 Frequency Table . . . 117
6.5 Standard Deviation Plot . . . 117
6.6 Small Programming Tasks . . . 118

Chapter 1

Introduction

One of the most effective approaches to describe, explore, and communicate data is visualisation. Data graphics are a tool to reason about quantitative information, and they can show unexpected things and raise new questions about the data.

Before data can be visualised it has to be stored in a consistent form that agrees with the semantics of the data, and it has to be prepared, for instance to deal with missing data. Data storage and preparation are the topics of Chapters 2 and 3.

Data visualisation is a science and an art. In the first place, a visualisation should be correct. It should not be misleading or distort the data. At the same time, it should be aesthetically pleasing. Chapter 4 introduces a powerful grammar of graphics. It is very flexible and has a logic for the mapping between the data and its representation, allowing a construction of plots. The aesthetic side of plots will be approached through the pioneering theory of Edward Tufte.

Chapter 5 shows how to apply plots and descriptive statistics to explore and describe data. Surrounding all the elements mentioned so far is programming. One does not have to be an expert programmer, but learning about programming allows to automate tasks and solve certain problems with ease.

The visual and descriptive approaches have their limits and that is where mathematical models have to be used. They describe and explain systems using mathematical objects and language, and can be used to make predictions inferences. Modelling and inferential statistics is not part of this course. Further, we largely ignore many technical aspects, e.g. aspects of data storage, security, governance and computational performance, and we mainly focus on concepts of data analytics.

These lecture notes are based on some material from a previous course and

have been rewritten and extended for D1. They can still be improved in many ways. Please report any mistakes you find to us. Suggestions and ideas for the next edition are also most welcome. We will provide the solutions for most of the exercises on Moodle.

Chapter 2

Databases

A big range of institutions, including the government, businesses, and the scientific enterprise generate and store data. Here are some examples:

- 300 hours of video are being uploaded every minute to Youtube.
- Twitter generates 500 million tweets/day.
- Wikipedia contains over 11 million articles.
- CERN's Large Hadron Collider generates 25 petabytes (one petabyte is about 10^{15} characters) per year of data.
- One high-throughput sequencing machine generates hundreds of gigabytes of sequence data (one gb is about 10^9 characters) in one week.

Of course, data is not only produced at extreme scales. Most companies have to store customer data or some inventory, and the world is populated by research groups storing their own data sets for every day work.

Flat-files

Arguably one of the simplest approaches to store data are flat-files. "Flat" means that there is no structure that indexes the data, and that there are no structural relationships between the entries in a file. As a consequence, flat-files can not be efficiently searched and manipulated. In particular, to work with a flat-file, it has to be read in its entirety into a computers memory. After a modification it has to be stored back in its entirety on a file-system.

A widespread example are comma-separated values (CSV) files that store a table:

IRIS.csv

```
5,0.194444444,3,0.416666667,1.6,0.101694915,0.2,0.041666667,setosa
5,0.194444444,3.4,0.583333333,1.6,0.101694915,0.4,0.125,setosa
5.2,0.25,3.5,0.625,1.5,0.084745763,0.2,0.041666667,setosa
5.2,0.25,3.4,0.583333333,1.4,0.06779661,0.2,0.041666667,setosa
4.7,0.111111111,3.2,0.5,1.6,0.101694915,0.2,0.041666667,setosa
4.8,0.138888889,3.1,0.458333333,1.6,0.101694915,0.2,0.041666667,setosa
5.4,0.305555556,3.4,0.583333333,1.5,0.084745763,0.4,0.125,setosa
7,0.75,3.2,0.5,4.7,0.627118644,1.4,0.541666667,versicolor
6.4,0.583333333,3.2,0.5,4.5,0.593220339,1.5,0.583333333,versicolor
6.9,0.722222222,3.1,0.458333333,4.9,0.661016949,1.5,0.583333333,versicolor
```

A further example is the biological community's use of flat-files to work with protein data, for example FASTA files for sequence data

```
>gi|121735|sp|P09488|GTM1_HUMAN Glutathione S-transferase Mu
  (GSTM1-1)(GTH4) (GSTM1A-1A) (GSTM1B-1B) (GST class-Mu 1)
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL
PYLIDGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNpef
eklkpkyleelpeklklYSEFLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPN
LKDFISRFEGLEKISAYMKSSRFLPRPVFSKMAVWGNK
>gi|232204|sp|P28161|GTM2_HUMAN Glutathione S-transferase Mu 2
  (GSTM2-2) (GST class-Mu 2)
MPMTLGYWNIRGLAHSIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL
PYLIDGTHKITQSNAILRYIARKHNLCGESEKEQIREDILENQFMDSRMQLAKLCYDPDF
EKLKPEYLQALPEMLKLYSQFLGKQPWFLGDKITFVDFIAYDVLERNQVFEPSCLDAFPN
LKDFISRFEGLEKISAYMKSSRFLPRPVFTKMAVWGNK
```

or sequence-entry files describing the function of a gene

```
ID   GTM1_HUMAN     STANDARD;      PRT;   217 AA.
AC   P09488;
DT   01-MAR-1989 (REL. 10, CREATED)
DT   01-FEB-1991 (REL. 17, LAST SEQUENCE UPDATE)
DT   01-NOV-1995 (REL. 32, LAST ANNOTATION UPDATE)
DE   GLUTATHIONE S-TRANSFERASE MU 1 (EC 2.5.1.18) (GSTM1-1) (HB SUBUNIT 4)
DE   (GTH4) (GSTM1A-1A) (GSTM1B-1B) (CLASS-MU).
GN   GSTM1 OR GST1.
OS   HOMO SAPIENS (HUMAN).
OC   EUKARYOTA; METAZOA; CHORDATA; VERTEBRATA; TETRAPODA; MAMMALIA;
OC   EUTHERIA; PRIMATES.
RN   [2]
RP   SEQUENCE FROM N.A.
RX   MEDLINE; 89017184.
RA   SEIDEGAERD J., VORACHEK W.R., PERO R.W., PEARSON W.R.;
RL   PROC. NATL. ACAD. SCI. U.S.A. 85:7293-7297(1988).
CC   -!- FUNCTION: CONJUGATION OF REDUCED GLUTATHIONE TO A WIDE NUMBER
CC       OF EXOGENOUS AND ENDOGENOUS HYDROPHOBIC ELECTROPHILES.
```

Problem with flat-files

There are two main problems with flat-files:

1. Structural relationships are missing
2. The files are redundant, i.e. the same information re-appears in a file or throughout a set of files.

Notice how the same term *setosa* reappears in the last column of the CSV file. The appearances are not connected. If you want to change *setosa* to *Iris setosa* you have to update every single instance. Imaging you have a set of FASTA sequence files and would like to get a list of all gene names appearing. Because structural relationships are missing, there is no easy way to obtain that information. Nothing says that the elements in the various descriptors are of the same type, e.g. that *GTM1_HUMAN* and *GTM2_HUMAN* are both gene names. Or notice the line starting with *OC* (denoting organism classification) in the sequence-entry file. The same information would appear in any sequence-entry file about a human gene, making it hard to update the classification.

Solution: data model and data base (DB)

The solution to these problems is abstraction with an appropriate *data model*. A data model determines the logical structure of the data. It describes explicitly the entities in the data (gene, amino-acid sequence, species) and how they are related (a gene has an amino-acid sequence, it has a function, it belongs to a species, one species has many genes).

Now, flat-files is not the way the data is stored by the providing resources. Typically, the underlying data is organised in a database (DBs). DBs are structured according to a DB model, which is one kind of data model. A DB model defines how data can be modified, searched, selected, and combined. They provide this a functionality without the user needing to worry how the data is physically stored.

In this course, you are going to learn about two DB models, the relational model and triples. The corresponding DBs are relational DBs (RDBs) and triple stores.

DB management system (DBMS)

DBs provide logical structuring of data and abstract away the physical structure. DB management Systems (DBMSs), in turn, implement the possibilities of the DB model. They are a software that interacts with the user, other applications, and the DB itself. They provide a concrete view on the abstract data model and allow the creation, querying, update, and administration of a DB. In the course, you will get in touch with a DBMS and use it to query a relational DB.

Data integration

The data in the world, or in one particular domain like biology, is not stored in one gigantic DB, but distributed in heterogeneous DBs. When we want to work

with data from various sources (for example, with sequence data *and* morphological data) and combine it, similar problems arises as above. The individual DBs are not linked and follow their individual data models. Structural relationships are missing. For instance, when are two entities in distinct DBs the same? Or how are the entities in one DB related to the entities in another DB?.

Solving these problems is the aim of data integration. The goal is to provide a unified view and access to a set of heterogeneous data sources. One way to tackle data integration is with the technology of the *Semantic Web*. In the last section, we will introduce the basics of the Semantic Web and conclude the chapter with a real world example of data integration. Ideally, by the end of the chapter, you should be able to understand Figure 2.3.

2.1 Principles of Relational Databases (RDBs)

The idea and theory behind RDBs were introduced in the 1970s by DaEdgar Codd, an English computer scientist. For his work, he was awarded the Turing Award in 1981 , which is recognized as the highest distinction in computer science, sometimes referred to as the "Nobel Prize of computing".

This section, in particular the example of mental health clinic DB, draws from a tutorial by Mariano Casanova published under a Creative Commons License.[1] We will cover the structure of RDBs and how to formulate simple queries.

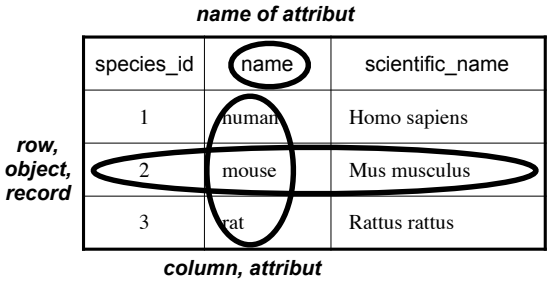
2.1.1 Structure of RDBs

A database consists of a group of tables that are interrelated. A table records information about objects that have the same characteristics. These characteristics are called attributes. One attribute in a table can reference a record in another table, allowing tables to connect information in ways that are very flexible and powerful when one later desires to retrieve this information.

Tables

Tables are organized in columns and rows. The columns represent different attributes that you want to store. You can have, for example, a table *Species* with columns representing the common name, scientific name, and some id (which we will use later in the example):

[1] *Base Tutorial: From Newbie to Advocate in a one, two... three!*, Mariano Casanova Available at: https://wiki.documentfoundation.org/images/0/02/Base_tutorial.pdf



Variable names

As a general rule, when you have to name variables in a system –for instance the name for a table, or an attribute, or a variable in R– you should conform to the following practices:

1. Start all variable names with a letter.
2. For subsequent characters use either letters, numbers or the underscore character.
3. Don't use a space between words. Instead separate them with the underscore.
4. Don't use special characters except for the underscore.
5. Use abbreviations, if needed, to help keep the length of variables names short.

The rules should ensure that you can let your data flow from one application to another.

Data types

The data that you will store in your tables will actually be stored as variables in your computer. Computers have different ways of storing data. In general, they trade memory or speed for accuracy: Computations that require more accuracy tend to be slower and use more memory. Each variable has one data type.

In general, computers handle numbers and characters differently. Some people are confused by the fact that numbers can be treated as characters also. e.g. '7' is a sign that represents the concept of seven. In that sense, it is stored in the same way as the signs '@' or '#' or the letter 'A'. As a character, the sign can not be operated mathematically, just as it would not make sense to calculate the sum of '@' and '#'. Phone numbers are good candidates for being stored as characters. Of course, any information that uses text is also stored as characters. Addresses need both, numbers and text, but we store them all as characters. Whether they are letters or numbers, when we are only storing

the signs (as opposed to the value for a number) we call them 'Alphanumeric characters'. We will not discuss here, how characters are stored internally, but look at numbers now.

Computers store information in binary form. The unit is called a 'bit' and can have one of two values (thus binary): either zero or one. Bits are organized in larger groups –usually eight bits– to form a Byte. In the following conceptual representation a byte is drawn as a box with eight spaces. Each space is a bit and can hold a zero or a one:

1	0	0	1	1	0	1	0

Since this is a binary numeral system, each box represents a power of two, in the same way our decimal number system assigns successive powers of ten from right to left:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Thus:

128	64	32	16	8	4	2	1

To represent the number five, you would need to 'activate' or indicate the numbers for four and one (because four plus one is five). So your byte representing five would look like this:

0	0	0	0	0	1	0	1

With a little bit of math you can figure out that you can represent all numbers from 0 to 255 (that is 256 numbers in total) using one byte. If you need to store numbers bigger than 255 then you need to add extra bytes.

Now, what if you need to store negative numbers, say -7? The solution for this has been to use the last bit to the left of your byte to represent the sign and use the other seven bits to represent the number. Because the last bit no longer represents the value 128 but instead works as a flag to indicate if the number is positive or negative, we can represent numbers form -128 to 127 with this arrangement. This type of bytes are called *signed*, while the bytes that only store positive numbers are called *unsigned* .

In general, numeric data variables are described by the number of bytes they use and whether they are signed or unsigned. At the least memory consuming side of number storage we have the Boolean numbers. A Boolean number is in fact just a bit, and we use it to store *yes/no* type of data. At the other end there are variables called *floating point numbers* (or just *Float*) that allow us to store numbers that have decimal places like 1.618033. We will not cover their internal representation here.

Another important type of variable is the *Date* type. They are used to store

calendar information like year, month, day, hour, minute, second and fraction of a second. There are several types, designed to be the most efficient in storing some or all of this information. The same is true for the Time type variable, which stores the time of the day: hour, minute and second.

The following table shows some main data types.

Data type	Size	Comment
Boolean	1Bit	Range: 0 and 1
Tinyint	1 Bytes	Unsigned, Range: 0–255
Integer	4 Bytes	$-2.14 \cdot 10^9$ to $2.14 \cdot 10^9$
Bigint	8 Bytes	$-2.3 \cdot 10^{18}$ to $2.3 \cdot 10^{18}$
Float	4 Bytes	Signed, $1.76 \cdot 10^{-38}$ – $3.40 \cdot 10^{38}$
Double	8 Bytes	Signed, $2.22 \cdot 10^{-308}$ – $1.79 \cdot 10^{308}$
VarChar	≤ 2 GB	Stores up to the specified length
Date	–	Month, day and year
Time	–	Hour, minute and second (in sec. since 1/1/1970)

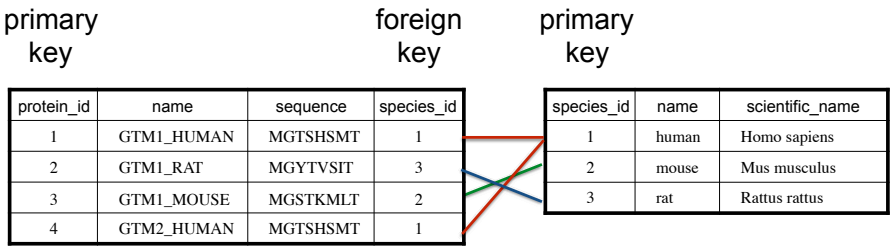
There are more data types and they can also differ from data base system to data base system.[2]

Relations: connecting tables

As a general rule, each table needs to cover one topic and one topic only. Above, we have introduced the table *species*. Here is a second table named *protein*.

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT	1
2	GTM1_RAT	MGYTVSIT	3
3	GTM1_MOUSE	MGSTKMLT	2
4	GTM2_HUMAN	MGTSHSMT	1

The column *species_id* indicates that the two tables are related as follows:

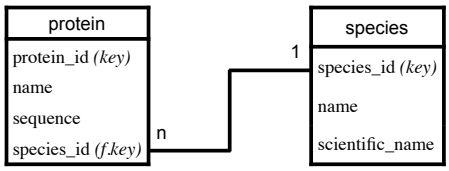


[2]The data types for Apache OpenOffice Base can be found at https://wiki.openoffice.org/wiki/Base/Data_Types

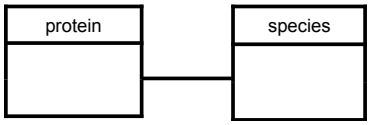
To achieve a link, each table needs a column (or combination of columns) to uniquely identify each record. Such a column (or collection of columns) is called a *primary key*. Further, a foreign key is needed in one of the tables. In our example, *protein* has a column *species_id* to link to *species*. Such a column (or a collection of columns) is called a *foreign key*. One species can have more than one protein. This means that the relation has a *one to many cardinality*.

UML

The presentation and design of databases is aided by a diagram protocol called UML, acronym for Unified Modeling Language. For our toy database it looks as follows:



The one to many cardinality is represented by "1" and "n" on the connecting line. One can also leave out details in a minimal representation:



Cardinalities

Above we have encountered the one to many relation or cardinality. There are three possible cardinalities.

- **one to one (1:1)** one occurrence of an entity relates to only one occurrence in another entity. For example, an employee is allocated a company car, which can only be driven by that employee.

- **one to many (1:n)** one occurrence in an entity relates to many occurrences in another entity. For example, an employee works in one department but a department has many employees.

- **many to many (m:n)** one occurrence in an entity relates to many occurrences in another entity. For example, an employee may work on several projects at the same time and a project has a team of many employees.

Many to many relationships are implemented with *intermediate tables*. You will encounter the case later in an exercise.

Normalisation

Normalisation is a formal way to ensure that our tables have a good structure, that is, that each table has the appropriate attributes. There exist at least ten normalisation rules, known as normal forms (NFs). We will cover here three important NFs, without going too much into detail.

First normal form (1NF):

1. The table contains only atomic values. An atomic value is a value that cannot be divided.
2. There are no repeating groups. A repeating group means that two or more columns repeat the same attribute.

The following two tables are not in NF1.

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

Student	Age	Subject1	Subject2
Adam	15	Biology	Maths
Alex	14	Maths	Physics
Stuart	17	Maths	Physics

The first table can be converted to NF1 as follows (the second one has a similar normalisation):

Student	Age	Subject
Adam	15	Biology
Alex	14	Maths
Stuart	17	Maths
Adam	15	Maths

This table needs two columns (Student, Subject) as primary key. Further, through NF1, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second normal form (2NF):

1. The table is in 1NF.
2. All non-key attributes are fully functionally dependent on the primary key (i.e. no partial dependencies on a concatenated key).

This means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails NF2. The table above fails NF2

because *Age* only depends on the *Student* column. To achieve NF2, one can split out the subjects into an independent table, and match the two tables up using the student names as foreign keys:

Student	Age
Adam	15
Alex	14
Stuart	17

Student	Subject
Adam	Biology
Alex	Maths
Stuart	Maths
Adam	Maths

Third normal form (3NF):

- It should not be the case that a non-prime attribute is determined by another non-prime attribute.

For example, consider a table with following fields:

Student_id	Student_name	Street	City	State	Zip
...	...	...	...	...	...

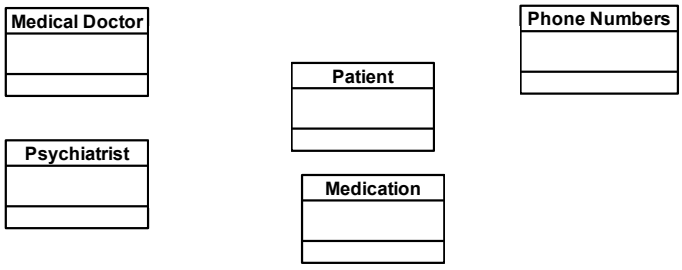
In this table Student_id is Primary key, but Street, City and State depend upon Zip. The dependency between Zip and other fields is called transitive dependency. To apply 3NF, we need to move the street, city and state to new table, with Zip as primary key.

Student_id	Student_name	DOB	Zip
...	...	...	...

Zip	Street	city	state
...	...	...	...

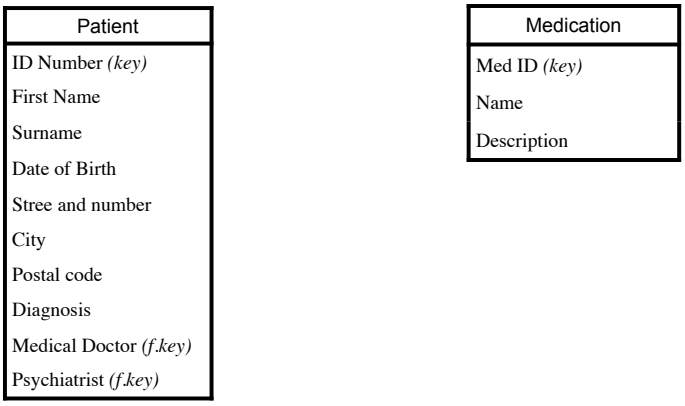
Exercise 2.1.1 *How would the number -257 be represented internally in the computer as data type* Integer*.*

Exercise 2.1.2 *Consider the following part of a mental health clinic DB.*



Add the relationships and cardinalities. (Do not peek at Figure 2.1.)

Exercise 2.1.3 *Here is a detailed view on the tables* patient *and* medication.



The relation between the two is many to many*. As mentioned in the section about cardinalities, tables that are related by* many to many *can not be connected directly. An intersection table is needed to connect them. Add an intersection table between* patient *and* medication*, name it* Patient/Medication *connect it to* patient *and* medication *using keys and add the attributes* Dosage, Start date *and* End date*. (Do not peek at Figure 2.1.)*

Exercise 2.1.4 *Now you can look at Figure 2.1. Study the entire design and discuss it with your peer(s). Would there be a problem with the design, if a therapist can also be a patient?*

Exercise 2.1.5 *There is one mistake "hidden" in Figure 2.1 concerning cardinalities. Can you find it?*

2.1.2 Querying RDBs

In this section you will query a toy database in *Apache Open Office Base*. *Apache OpenOffice* is an open-source office suite, i.e. a collection of bundled software, including a word processor, a spreadsheet, a presentation program and more. It also contains *Base*, a relational database management system (analogous to Microsoft Access).

Base has a lot of functionalities, for instance, it can serve as a front-end to a number of different database systems. Here we will work with a *personal use database* stored in the file *toyGeneDB.odb*, which you can find on Moodle. You will work through a number of tasks, first creating queries using Base's design view, and then having a peek into the query language SQL.

Task 1

a. Load the database toyGeneDB.odb in OpenOffice. Double clicking on the file will not work on some systems; instead, it has to be opened from

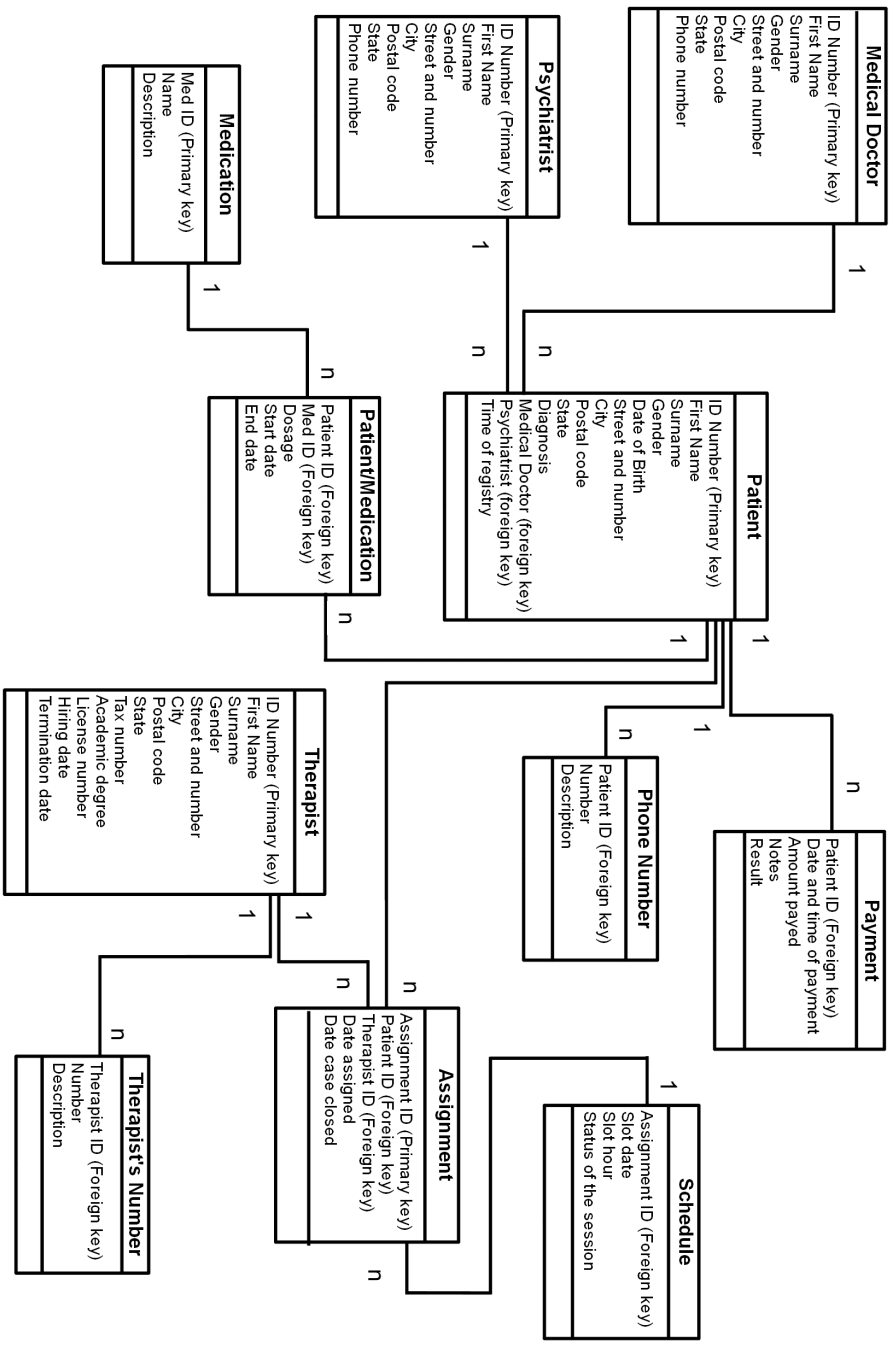


Figure 2.1: Diagram of a mental health clinic DB

within OpenOffice (as described below).
Hint: 1. Start OpenOffice; 2. Click on *Database*; 3. Check the radio-button *Open an existing database file*; 4. Click *Open* and navigate to the location of *toyGeneDB.odb* on your computer.

b. There are two tables in the DB (*protein* and *species*). Look at them.
Hint: 1. Double-click the table and look at the content. 2. Go back to the main window, right-click the table and select *edit*.

c. What is their cardinality?

d. How are the tables related?
Hint: In the menu-bar, click on *Tools* and then *Relationships*.

In the following tasks you will create queries using the *Design View*. The solutions to all the queries can be found in the database *toyGeneDB_withQueries.odb*.

Task 2

a. **Query1:** Select the column *name* from the table *protein* (hereafter, referred to as *protein.name*).

b. Sort the entries in *protein.name* in descending order by hand.

c. **Query2:** Select the columns *name* and *sequence* from the table *protein*, sort the result in descending order of the names (use *Sort* in the design view).

d. **Query3:** In the *species* table, show the *name* and *scientific_name* of the *rat*.
Hint: use *Criterion* in the design view.

e. **Query4:** Do the same as in Query3, but only display the *scientific_name*

f. **Query5:** In the *species* table, show the *name* and *scientific_name* of *rat* and *human*
Hint: use *Criterion* and *Or* in the design view.

g. **Query6:** Select the protein names and sequences for proteins that contain the subsequence *SHS*.
Hint: Insert *LIKE '*SHS*'* in the appropriate *Criterion*-field.

h. **Query7:** Select the protein names and sequences for proteins that contain the subsequence *SHS* or the subsequence *GY*.

i. **Query8:** Show all the *human proteins*.
Hint: In the design view form, you need on column for the species table (to insert *human* in *Criterion*), and one column for the protein table (to select the protein name).

j. **Query9:** Count the number of human proteins.
Hint: i. Build on Query 8; ii. Make sure to *disable visibility* of the species name; iii. Use the function *Count*.

Task 3

Internally, the queries generated by the GUI are translated to SQL. SQL, or Structured Query Language, is a language used to create relational databases and manipulate data in them. The general structure of a basic query in SQL looks as follows:

```
SELECT column_name
FROM table_name
WHERE column_name operator;
```

The SELECT/FROM statement is used to select data from a database. The result is stored in a result table. The WHERE clause is used to extract records that meet a specified criterion. This may look a bit abstract, but will hopefully make sense when you look at actual examples.

In this exercise you will look at the SQL-queries resulting from the previous tasks and modify some of them. To access the SQL code of a particular query, right-click on the query and select "Edit in SQL view".

a. **SQL-Query1:** Modify Query1 to show *species.scientific_name*.

b. **SQL-Query2:** Modify Query2 to sort in descending order (substituting *DESC* by *ASC*).

c. **SQL-Query3:** Modify Query3 to show the scientific name (only) of human.

d. **SQL-Query4:** Modify Query5 to show the sequence of *GTM1_HUMAN* and *GTM2_HUMAN*.

e. **SQL-Query5:** Modify Query6 to show results of proteins that contain the subsequence *KM*.

f. Study Query8 to see in SQL how the two tables are connected over *protein.species_id* and *species.ID*.

2.2 Introduction to the Semantic Web

The World Wide Web (WWW) was invented by Sir Tim Berners-Lee in 1989. The key technology of the original web, from an end user's point of view, is the hyperlink. A user can click on a link and immediately go to the document identified in that link.

The great advantage of the WWW (or Web 1.0) is that it abstracts away the physical storage and networking layers involved in information exchange between two machines. This enables documents to appear to be directly connected to one another. Click a link and you are there, even if that link goes to a different document on a different machine on another network on another continent.

In the same way that Web 1.0 abstracts away the network and physical layers, the *Semantic Web* (also referred to as Web 3.0, the Linked Data Web, the Web of Data)[3] abstracts away the document and application layers involved in the exchange of information. The Semantic Web connects facts, so that rather than linking to a specific document or application, you can instead refer to a specific piece of information contained in that document or application.

The word semantic itself implies meaning or understanding. As such, the fundamental difference between Semantic Web technologies and other technologies related to data (such as RDBs or the WWW itself) is that the Semantic Web is concerned with the meaning and not the structure of data.

The Semantic Web consists primarily of three technical standards:

1. **Resource Description Framework (RDF)**: The data modelling language for the Semantic Web. Semantic Web information is stored or represented in the RDF.

2. **SPARQL** is pronounced "sparkle" (a recursive acronym for *SPARQL Protocol and RDF Query Language*). It is the query language of the Semantic Web and specifically designed to query data across various systems.

3. **Ontologies and the Web Ontology Language (OWL)**: Ontologies are a formal way to describe taxonomies and classification networks, essentially defining the structure of knowledge for various domains: the nouns representing classes of objects and the verbs representing relations between the objects. OWL is the knowledge representation language of the Semantic Web, and used to describe Ontologies. It is based on RDF.

In this course, we will not go into technical details but rather introduce RDF and Ontologies from a conceptual point of view. SPARQL will not be covered. You have had a peek at SQL. Think of SPARQL as a similar language used to query RDF data. In the last section, we will outline in a real example how the Semantic Web is used to integrate heterogeneous resources.

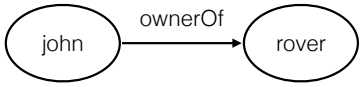
[3] You may be wondering what the Web 2.0 is. Web 2.0 does not refer to any technical specification, but to changes in the way Web pages are designed and used. A Web 2.0 website may allow users to interact and collaborate with each other in a social media dialogue as creators of user-generated content in a virtual community (think of Facebook, or video sharing on Youtube), in contrast to the Web 1.0 websites where people were limited to the passive viewing of content. Whether Web 2.0 is substantively different from prior Web technologies has been challenged by Tim Berners-Lee. *(Source: Wikipedia)*

This section uses examples and parts of the text from the the following resources. The Shakespeare example is from a colleague's (Nives Skunca) presentation held at a *Scientific Databases* course at ETH.[4] The example about John and Rover is borrowed from a video tutorial by Rob Taylor.[5]

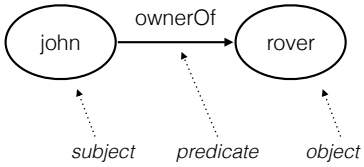
2.2.1 RDF

RDF is the data model of the Semantic Web. That means that all data in Semantic Web technologies is represented as RDF. If you store Semantic Web data, it is in RDF. If you query Semantic Web data (using SPARQL), it is RDF data. RDF is not like the tabular data model of relational databases. Instead, RDF is a graph with nodes and directed edges.

An RDF graph is composed of statements. Each statement asserts a fact about a resource and is called a *triple*. Here is a triple for the statement "John is the owner of Rover":



A triple has three parts: a subject, an object and a predicate. In our example *john* is the subject, rover the object, and *ownerOf* the predicate.

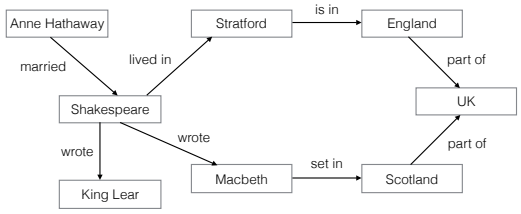


An RDF database is often called a triple store for this reason.

2.2.2 Merging data

Consider the following triples and the corresponding graph (both representing a triple store):

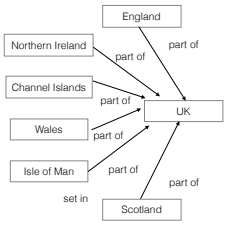
Subject	Predicate	Object
Shakespeare	wrote	Macbeth
Shakespeare	wrote	King Lear
Anne Hathaway	married	Shakespeare
Shakespeare	lived in	Stratford
Stratford	is in	England
Macbeth	set in	Scotland
England	part of	UK
Scotland	part of	UK



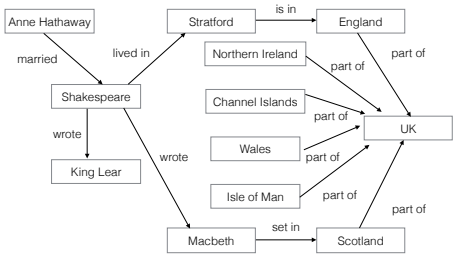
[4] http://www.cbrg.ethz.ch/education/SDB/
[5] RDF Beginners Guide, https://vimeo.com/9848513

Assume now, we would like to include the following information in our model:

Subject	Predicate	Object
Scotland	part of	UK
England	part of	UK
Wales	part of	UK
Northern Ireland	part of	UK
Channel Islands	part of	UK
Isle of Man	part of	UK



UK in both graphs refers the same thing, so we can merge the triples:



In our example, it is "clear" that UK is the same thing in both graphs. A general question is: When is a node in one graph the same node in another graph? The answer is, if they have the same URI. In the following section we will introduce URIs.

2.2.3 RDF Schema and URIs

In the previous section, we have represented triples graphically. In actual specifications a notation is used, which looks as follows:

```
<john> <owner> <rover>
```

The brackets indicate that the things in brackets are references. We are using a simplified notation here. In reality the references are URIs. Below we will take URIs up again. First, however, we are going to introduce a further level of abstraction on top of RDF. This abstraction is provided by *RDF Schema (RDFS)*.

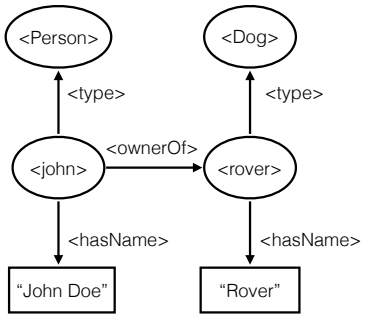
RDF Schema

RDF Schema is an extension of basic RDF. It provides basic elements for the description of ontologies. RDF creates a graph structure to represent the data; RDFS gives guidelines how to use the graph structure in a disciplined way. Consider the following extension of our example:

```
<john> <type> <Person>
<rover> <type> <Dog>
```

```
<john> <hasName> "John Doe"
<rover> <hasNamer> "Rover"
<john> <owner> <rover>
```

We have introduce three new things here: *classes, instances, literals.* Classes are a classification of a certain kind of things. Here we say that $< john >$ is an instance of the class $< Person >$, and $< rover >$ an instance of the class $< Dog >$. A literal is a constant value and has a data type (see section *Data types* above). Here is a graphical representation of the RDFS specification:



URIs (and IRIs)

As mentioned above, we have been using a simplified notation for the references in brackets. In proper RDF the references are Uniform Resource Identifiers (URIs), or IRIs[6]. An URI is a unique identifier to identify a resource over a network, in particular, over the WWW. One familiar example of URIs are URLs, i.e. web address, like the ones you type in your browser. So the first triple from the RDFS code above

```
<john> <type> <Person>
```

would rather look something like this:

```
<http://example.org/example#john>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://example.org/example#rover>
```

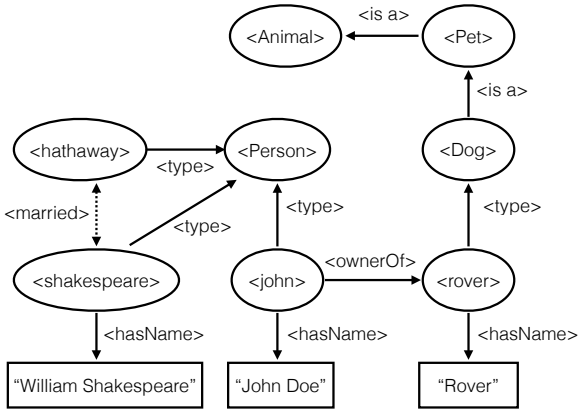
As a consequence, each element of a triple can be stored anywhere on the internet an be accessed by its URI. We will not go into more detail here about URIs.

2.2.4 Ontologies, OWL, Inference

Ontology is the philosophical study of the nature of being, becoming, existence or reality, and the basic categories of being and the relations between beings.

[6]The Internationalized Resource Identifier (IRI) was introduced in 2005 as a new internet standard to extend upon URI. While URIs are limited to a subset of the ASCII character set, IRIs may contain characters from the Universal Character Set, including Chinese or Japanese kanji, Korean, Cyrillic characters, and so forth. *Source: Wikipedia*

The term Ontology, as employed in the context of the Semantic Web (and in general in computer sciences) can be described as a practical application of the philosophical ontology. Ontologies are a formal way to name entities and the interrelationships of the entities that exist in a particular domain of interest. The formal way is realized by a language, such as RDFS, which we have encountered above with a simple example. Here is an ontolgoy extending the example:



In the context of RDFS we have seen that RDF can be extended with further concepts, such as classes and instances. The Web Ontology Language (OWL), in turn, is also based on RDF and extends RDFS. OWL offers constructs, for instance, to express that a certain relation is symmetric, or that an entity (such as the emotion *pleasure*, see Figure 2.2) is an intersection of other entities (e.g. of *aesthetic pleasure, sexual pleasure* or *schadenfreude*).

Such constructs can be used to reason logically about things, and to infer new statements. Consider, for example, the link between $< shakespeare >$ and $< hathaway >$. The double-arrow indicates that $< married >$ is a symmetric relation. From the symmetry, and from the fact that Hathaway was married to Shakespeare, we can deduce that Shakespeare was married to Hathaway. From the ontology we can also deduce that Shakespeare could have owned a dog, that pets can be owned, or that Rover is an animal.

2.2.5 OBO

The bioinformatics community uses various domain specific ontologies that are based on manual integration approaches by human experts. Many ontologies are included in the OBO Foundry (Repository of Life-Science Ontologies [6]) a collaborative effort, which includes about 140 ontologies and is now considered a gold standard. Specifically, the OBO Foundry has established principles for ontology development and evolution, with the aim of maximizing cross-ontology coordination and interoperability.

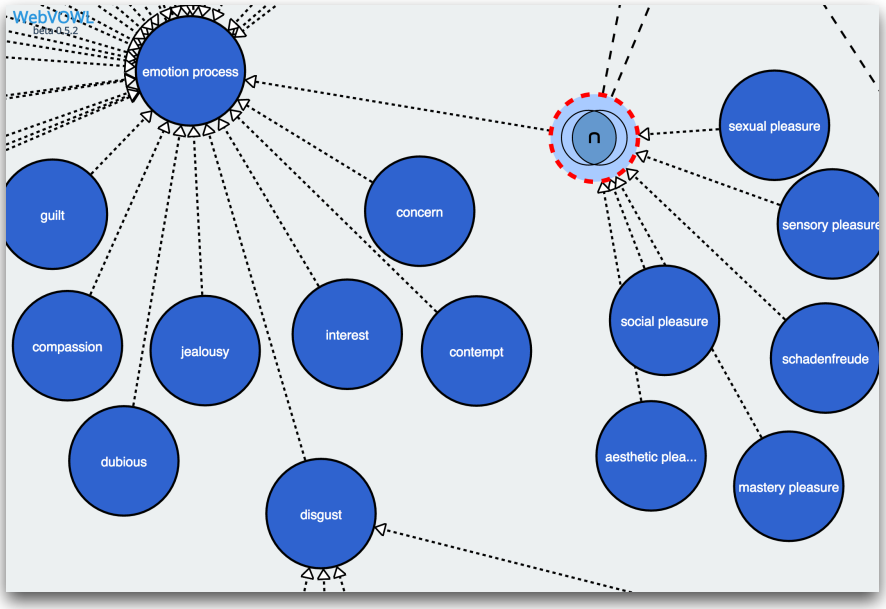


Figure 2.2: Excerpt from the Emotion Ontology. The node highlighted with a red, dashed circle stands for the concept *pleasure* and is an intersection of the concepts pointing to it. The figure was generated with *WebVOWL: Web-based Visualization of Ontologies* at http://vowl.visualdataweb.org/webvowl/.

Examples are the Gene Ontology (concepts used to describe gene function, and relationships between these concepts), Uberon (cross-species anatomy), or the Confidence Information Ontology (CIO). The CIO ontology can encode, for example, whether a sequence alignment corresponds to a weak similarity over part of the protein, or to 99% identity over the whole length of the protein. The example above about *pleasure* being the intersection of other entities stems from the Emotion Ontology (see Figure 2.2).

2.2.6 RDF and RDBs

An RDB can be mapped or converted to RDF. Remember our protein and species example from the RDB section:

primary key

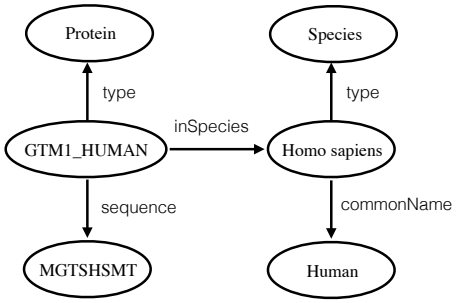
foreign key

primary key

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT	1
2	GTM1_RAT	MGYTVSIT	3
3	GTM1_MOUSE	MGSTKMLT	2
4	GTM2_HUMAN	MGTSHSMT	1

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

The first record of each table and their connection could be expressed as follows in RDF:



RDB data can integrated in two ways with RDF:

1. **Native triple store**: the RDB is fully converted to a triple store. The store is implemented from scratch as RDF exploiting the RDF data model to efficiently store and access the data. The example above is such a conversion.

2. **RDB backed**: the RDB is kept to store the data, and a triplestore is built by adding an RDF specific layer to the RDB. The layer models the schema of the RDB. This approach is suitable when a resource has established dependences with users and resources. We will encounter such an example in the section *Real example of data integration.*

Exercise 2.2.1 *Convert the entire toy protein/species RDB to an RDF triple store. Hint: extend the example above.*

Exercise 2.2.2 *Select an OBO ontology and display and explore it with WebVOWL. Hints:*

- *Visit http://www.obofoundry.org/*
- *Choose an ontology that you would like to explore and click on its name. For example, to choose the* Emotion Ontology *you would click on* mfoem.
- *You get to a new page. On the right side of the page copy the link next to* PURL. *It is an IRI. For the* Emotion Ontology *the IRI is: http://purl.obolibrary.org/obo/mfoem.owl*

- *To display the Ontology with WebVOWL visit http://visualdataweb.de/webvowl/ and follow the menu (Ontology → Custom Ontology).*
- *You should now be able to graphically explore the ontology.*

Exercise 2.2.3 *Choose any domain that interests you and design your own small ontology, displayed as a graph with nodes and edges. Be creative.*

2.2.7 Real example of data integration

Rapid progress in DNA sequencing and other high-throughput technologies are transforming biological sciences into a data-intensive discipline. Thanks to an open data culture in biology, much of this data is publicly available in a wide array of databases. Switzerland is at the forefront of bioinformatics resource provision, being home to several of world-leading databases (including Swiss-Prot, neXtProt, OMA or Bgee) federated via the SIB Swiss Institute of Bioinformatics.

One major potential and promise of the semantic web lies in the simultaneous mining of multiple sources of data. However, while these databases can be individually queried via, they remain largely organized as independent data sources with limited references to each other.

In a collaborative effort, the Zurich University of Applied Sciences, University of Lausanne, and Swiss Institute of Bioinformatics are developing a Semantic Web architecture to connect the databases and to *integrate the data* from the heterogeneous resources. Figure 2.3 shows the architecture and explains its parts. Are you able to understand the architecture with the knowledge gained in this chapter (of course, without the underlying technical details)?

The architecture has three layers denoted by A, B, C (in red circles) and cross-references (green arrows).

A. This layer is the Base Data. It represents the DBs that we would like to integrate. In this example there are two DBs. On the left is OMA. OMA provides orthology information and is a triple store.

B. This part is called the Data Model Layer. It provides a consistent interface to the base data and is expressed in RDFS. In the case of the OMA triple store, the model layer adds *classes* to describe the structure of the base data. The RDB Bgee is *RDB backed*.

C. This layer, called Integrated Domain Ontology. It is one big ontology composed of OBO ontologies and expressed in OWL. It connects to the Data Model layer (symbolised by the green bands), extends the vocabulary of the Data Model Layer, and allows reasoning about the base data.

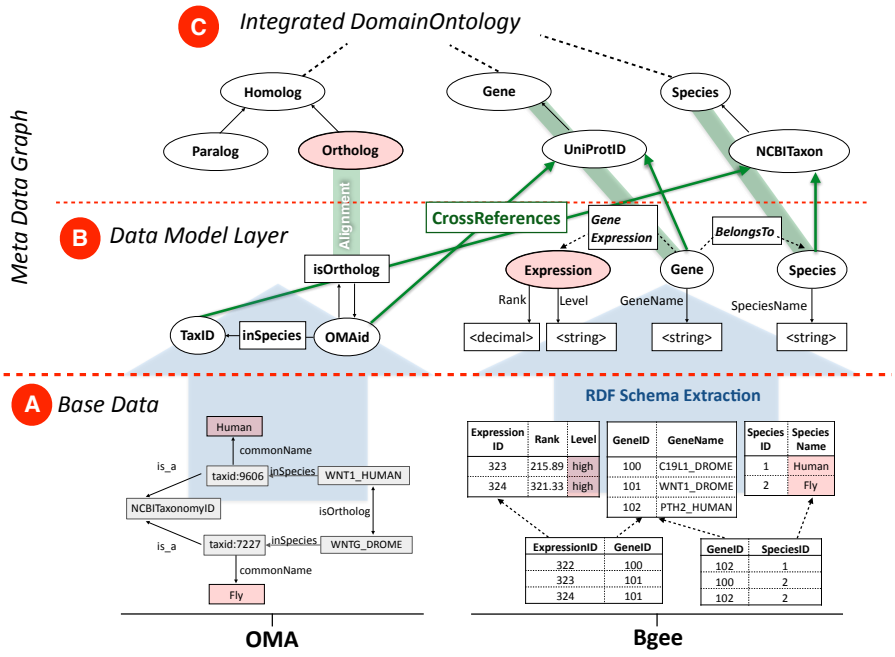


Figure 2.3: Example of a semantic web architecture, see text

The green arrows represent **cross-references**. The entities stored or referred to in the base data (e.g. actual sequences, orthologous groups, anatomical parts, species etc.) do not necessarily share the same database identifiers and keys. Therefore, one has to explicitly establish a mapping about the things that are equal. This can be achieved by mapping the entities to one established referencing system. For species, for instance, this can be the NCBI taxonomy, a curated classification and nomenclature for all of the organisms appearing in public sequence databases. In the architecture the referencing is achieved by linking to the NCBITaxon ontology, which is an automatic translation of the NCBI taxonomy into OWL (so that each species has an unique URI).

Finally, the layers B and C make up the **Meta Data Graph** (or Meta Data Layer, as opposed to the instances in the Base Data Layer). The Meta Data Graph is fully expressed in RDF and can, thus, be queried with SPARQL.

Chapter 3

Data preparation in R

Statistical theory typically assumes that data are in a consistent and correct state for analysis. In practice, however, prior to any statistical operation a lot of time is spent preparing the data (80% of data analysis, according to [2]). Data preparation includes a number of tasks, for instance, outlier checking, handling missing data, detecting inconsistencies (e.g. a man can not be pregnant) proper parsing of formats (e.g. dates). In this course, we focus on two important tasks: data tidying, and handling missing data.

3.1 Data Tidying

The concept of data tidying, that is *structuring data to facilitate analysis*, has been developed by Hadley Wickham. He is the developer of well known R packages and the Chief Scientist at RStudio. In Chapter 4 we will introduce *ggplot*, one of his packages for data visualisation. The present section is based on his main publication on data tidying [10]. Most examples are borrowed from a book by Wickham, published under a Creative Commons license and available online [11].

3.1.1 Prerequisites

Before we start, we have to install the package *tidyverse*. Start RStudio and run the command:

```
install.packages("tidyverse")
```

The line of code will download a number packages from CRAN[1]. To use the contents you have to load the libraries:

```
library(tidyverse)
```

[1]The Comprehensive R Archive Network is a network of servers around the world that store identical, up-to-date, versions of code and documentation for R.

```
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages
    ----------------------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

In conjunction with the packages a collection of datasets (from [11]) is obtained, for example:

```
table1
# A tibble: 6 ? 4
#       country  year  cases population
#         <chr> <int>  <int>      <int>
#1 Afghanistan  1999    745   19987071
#2 Afghanistan  2000   2666   20595360
#3      Brazil  1999  37737  172006362
#4      Brazil  2000  80488  174504898
#5       China  1999 212258 1272915272
#6       China  2000 213766 1280428583
```

We will use the datasets later in the chapter. Note that the table is of type *tibble*.

3.1.2 Data frames of type *tibble*

A *tibble* is a data frame. At any time, you can convert a *tibble* to R's classic data frame and vice versa:

```
dfr = data.frame(v1=c("a", "b", "c"), v2=c(1, 2, 3))
tib = as_tibble(dfr)
dfr2 = as.data.frame(tib)
```

In most cases tibbles behave like R's classic data frames. One key difference between the two is printing. In contrast to classic data frames, tibbles report the type of each column:

```
print(dfr)
print(tib)
print(dfr2)
```

The refined *print()* method for tibbles shows only the first 10 rows, and all the columns that fit on screen. This makes it easier to work with large data. Furthermore, you can control the number of rows (n) and the *width* of the display. Note that *width=Inf* will display all columns.

```
# Data frame with normally distributed random sample:
dfr = data.frame(matrix(rnorm(20), ncol=30, nrow=30))
print(dfr)
tib = as_tibble(dfr)
print(tib, n=10, width=nf)
print(tib, n=10, width=50)
print(tib[c(10,11)], n=10)
```

Alternatively, you can also work with the *View()* method:

```
View(dfr)
View(tib)
```

Note that is is possible for a data frame to have column names that are not valid R variable names, aka non-syntactic names. For example, they might not start with a letter, or they might contain unusual characters like a space. To refer to these variables, you need to surround them with backticks ‘:

```
dfr1 = data.frame(v1=c("a", "b", "c"), v2=c(1, 2, 3))
# Gives Error:
dfrE = data.frame(1=c("a", "b", "c"), 2=c(1, 2, 3))
# Works fine:
dfr2 = data.frame(‘1‘=c("a", "b", "c"), ‘2‘=c(1, 2, 3))
```

Exercise 3.1.1 *Experiment with the parameter* n *of the* print() *method.*

Exercise 3.1.2 *Experiment with the parameter* width.

3.1.3 Structure of datasets

Many datasets are tables made up of rows and columns. Table 3.1 provides a toy example of a some data comparing two drugs. It has two columns and two rows, all the rows and columns are labelled.

	Drug A	Drug B
Peter Graf	–	2
Andrea Bauer	16	11
Stefan Federer	3	1

Table 3.1: Typical representation of data

There are other ways to structure the same data, like for example in Table 3.2, where the rows and columns have been transposed. The words *row*, *column*, and *label* can be used to talk about the layout of the data. In addition, we need a way to talk about the meaning, that is the *semantics*, of the data.

	Peter Graf	Andrea Bauer	Stefan Federer
Drug A	–	16	3
Drug B	2	11	1

Table 3.2: Same data as in Table 3.1 represented differently

3.1.4 Meaning of datasets

A dataset is a collection of *values*, most often numbers or strings. There are two aspects to a value.

1. A value belongs to a *variable*. A variable stands for a characteristic that can be measured or counted. Age, sex, eye colour, temperature, duration are examples of variables.

2. A value also belongs to an *observation*. An observation contains all the values that have been measured on the same entity across the variables. A person, a day, a religion are examples of entities.

name	drug	result
Peter Graf	A	–
Andrea Bauer	A	16
Stefan Federer	A	3
Peter Graf	B	2
Andrea Bauer	B	11
Stefan Federer	B	1

Table 3.3: Same data as in Table 3.1. Observations are organised in rows, variables in columns.

Table 3.3 shows the same data as Tables 3.1 and 3.2, but makes the values, variables, and observations explicit. There are three variables, six observations, and 18 values. The variables are:

- *person*, with values Peter, Andrea, Stefan
- *drug*, with values A, B
- *result*, with 5 numerical values and one missing value

In Section 3.2 we will discuss how missing values can be treated.

3.1.5 Tidy data

Tidy data is a way to map the meaning of a dataset to its structure and defined as follows:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each value is placed in its own cell.
4. Each type of observational unit forms a table.

Any other arrangement we will call *messy data*. Note that Table 3.3 is tidy, Tables 3.1 and 3.2 are messy. To exemplify the last point in the definition, we could imagine an additional observational unit recording the age, weight, and sex of the three individuals. Such a dataset would be placed in its own table. Note that tidy data is modelled according to database normalisation.

3.1.6 Tidy works well with R

One might think that the three tables are interchangeable, after all they represent the same information. However, tidy data is much easier to work with in R than messy data. Tidy works well with R, because R is a vectorised language. This means that the data structures are composed of vectors, and the functions operating on the data structures are optimised for vectors. A key example is the data structure *data frame*. A data frame is a list of vectors, displayed as a table, where each vector corresponds to a column. Consequently, for tidy data, each variable resides in its own vector.

Exercise 3.1.3 *With the tidyverse package you have downloaded the following tables: table1, table2, table3, table4a, and table4b. They show the same dataset, but each table structures it differently. The data comes from the World Health Organisation, and records counts of confirmed tuberculosis cases. Inspect the tables and describe how the variables and observations are organised. Which one is the tidy table?*

Exercise 3.1.4 *Using R's vector arithmetic and the tidy table, compute for each country the rate of tuberculosis cases (as cases per 100'000 inhabitants). Add the column to the tidy table and afterwards remove it again.*

In the following, we will look at common problems with messy data.

3.1.7 From messy to tidy

The five most common problems with messy data are:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Multiple variables are stored in both one row and one column
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

In the following, we will focus on the first three points.

Column headers are values, not variable names

In some datasets the column names are not names of variables, but values of a variable. This is the case for *table4a*. The column names 1999 and 2000 represent values of the year variable, and each row represents two observations, not one. To tidy a dataset like this, we need to convert those columns into a new pair of variables. This can be achieved with the function *gather()*, which needs the following information:

- The column names that represent values, not variables. Here, they are 1999 and 2000.
- A name for the variable whose values form the column names. Here, we choose the name *year*.
- The name of the variable whose values are spread over the cells. Here, it is the number of tuberculosis cases, which we denote by *cases*.

The call to *gather()* is:

```
gather(table4a, `1999`, `2000`, key="year", value="cases")
```

Note how the "variables" are surround with backticks. This is needed, because the names start with a number.

Multiple variables are stored in one column

In this section we are concerned with *table2*. Note how the column *count* contains both the count of the tuberculosis incidents and the count of the population (i.e. population-size). The corresponding variable names are recorded in a column denoted with *type*. As a consequence, each observation is scattered across multiple rows. This can be fixed with the function *spread()*. It needs the following information:

- The column with the variable names (here, *type*). This column is passed by the parameter named *key*.
- The column that contains values forms multiple variables (here, *count*), passed by the parameter named *value*.

The call to *spread()* is:

```
spread(table2, key="type", value="count")
```

Exercise 3.1.5 *Why does spreading this tibble fail? How could you add a new column to fix the problem?*

```
people = tibble(
   name = c("Phillip Woods", "Phillip Woods", "Phillip
       Woods", "Jessica Cordero", "Jessica Cordero"),
   key = c("age", "height", "age", "age", "height"),
   value = c(45, 186, 50, 37, 156)
)
```

Multiple Variables are stored in both one row and one column

A complicated form of messy data occurs when variables are stored in both one row and one columns. In *table3* the column *rate* contains the two variables *cases* and *population*. To fix the problem we use the function *separate()*. As arguments, it takes the name of the column to separate, and the names of the columns to separate into. Further, you can specific a character to separate the column; here it is "/".

```
table3Tidy = separate(table3, rate, into=c("cases",
    "population"), sep="/")
```

If you look at the column types, you will notice that *case* and *population* are character columns. This is the default behaviour in *separate()*, it leaves the type of the column as is. Here, it is not very useful as those are really numbers. We can ask *separate()* to try and convert to better types using *convert = TRUE*:

```
separate(table3, rate, into=c("cases", "population"),
    sep="/", convert = TRUE)
```

Exercise 3.1.6 *Instead of using the argument* convert = TRUE *in the call to* separate(), *columns can also be converted explicitly using R's data type conversion with* as. *Apply it to convert the same two columns as above to the type* integer *in* table3Tidy.

3.1.8 Concluding remarks

In practice raw data is rarely tidy. Getting your data into the tidy format requires some upfront work, but that work pays off in the long term. Once you have tidy data, you will spend less time converting data from one representation to another, allowing you to spend more time on the analytic questions at hand, in particular, because the format has become integral to the R cosmos.

Here, you have learned about a few important functions from the *tidyverse* package. It contains more functions to tidy datasets. If you would like to learn about them, you can consult the Wickham's book (available online [11]).

Note that *messy* is a pejorative term and a such an over-simplification of the situation. There are well motivated and useful non-tidy data-structures. Such

alternative representations may have advantages in terms of space and time performance. To this effect, specialised fields have come up with their own standards for storing data. Non-tidy data, however, is outside the scope of this course and we will not treat it here any further.[2]

3.2 Missing Data

Missing data is a common problem in empirical research. It is itself a subject of research with a rich body of literature, including statistical methods to substitute missing values, a process called *imputation*. An in-depth treatment of imputation methods is beyond the scope of this course. Instead, the goals are to characterise the problem of missing data, to introduce the concept of imputation, and to apply a state of the art R package.

This section draws from the book *An introduction to data cleaning with R* (available online at CRAN [3]), from a review article [7], and from a tutorial published under a Creative Commons License at *Datascience+*[3]. Datascience+ is a collection of R tutorials for data scientists.

We will use the popular imputation package *Mice* [1]:

```
install.packages("mice")
library("mice")
```

Furthermore, we will need a function from the following package:

```
install.packages("missForest")
library("missForest")
```

3.2.1 Classifying missing data

Missing data can occur for different reasons. In an experimental study a researcher may be unable to collect an observation. She may become sick or drop a test tube. The equipment can fail. The weather precludes an observation in a field experiment. Data may be missing due to an data entry error. Subjects in longitudinal studies may drop out because they do not like the treatment, or they die. Surveys suffer missing data when participants skip an item.

Two main types of missing data can be distinguished [4]:

- *MAR*: missing at random. In this case, conclusions based on data with missing values should not differ from conclusions based on complete datasets.

[2] If you would like to find out more about non-tidy data, you can read the following blog post: http://simplystatistics.org/2016/02/17/non-tidy-data/

[3] http://datascienceplus.com/imputing-missing-data-with-r-mice-package/

- *MNAR*: missing not at random is a more serious issue. In this case, it might be wise to check the data gathering process further. The nature of the pattern needs to be understood before one can interpret the results correctly. An example of MNAR is men failing to fill in a depression survey because of their level of depression.

Note that if the missing data pattern is not random, then there is no statistical approach to solve the problem. Thus, techniques for dealing with missing observations assume that the pattern of data loss is random[7].

3.2.2 Handling missing data

In the following we are going to work with the *Iris dataset*. It is included in standard R and can be accessed by the name *iris*. First, we will create an iris data frame with 10% randomly missing values using the prodNA function from the missForest package:

```
iris.mis = prodNA(iris, noNA = 0.1)
```

We want to focus on numerical values and remove the categorical variable *Species*.

```
iris.mis = subset(iris.mis, select=-c(Species))
```

The mice package provides a nice function *md.pattern()* to get an understanding of the pattern of missing data.

```
md.pattern(iris.mis)
```

Exercise 3.2.1 *Try to understand the output of* md.pattern().

Deleting data

If the amount of missing data is very small relatively to the size of the dataset, then leaving out the few samples with missing features may be the best strategy in order not to bias the analysis. The rows in *iris.mis* containing missing values can be removed as follows:

```
iris.del = na.omit(iris.mis)
```

However, leaving out available data points deprives the data of some amount of information. It can substantially lower the sample size, leading to a lack of power. This is especially true if there are many variables involved in the analysis, each with data missing for a few cases. Depending on the situation, you may want to look for other fixes than deleting potentially useful data points from your dataset. Imputation is such a fix.

Imputation of the Mean

One of the simplest imputation methods is to replace the missing observations of a particular variable x by the mean $\bar{x}$ of the observed values of x.

Exercise 3.2.2 *Apply this approach to the variables* Sepal.Length *from* iris.mis.

Exercise 3.2.3 (Advanced) *Write a function that takes a data frame as an argument, performs* Imputation of the Mean, *and returns a data frame with imputed values.*

Regression Imputation

With *Imputation of the Mean* all the missing values are replaced by the same number. This may lead to underestimates of the variability in the data. In contrast, *Regression Imputation* computes a number for each missing value. The approach works in two steps. Suppose we have the variables $x_1, x_2, \ldots x_k$, and x_1 has missing values.

1. Estimate the relationship between x_1 and $x_2, \ldots, x_k$, for instance, by linear regression.
2. Use the regression model to predict the missing values in x_1.

The following code gives an idea, how this can be done in R "by hand". Try to run and understand the code.

```
# Regression: "Sepal.Length" on "Sepal.Width",
    "Petal.Length", "Petal.Width"
iris.mis.lm = lm(Sepal.Length ~ Sepal.Width + Petal.Length
    + Petal.Width, data=iris.mis)

# Extract the rows from iris.mis where Sepal.Length is NA
tmp = iris.mis[is.na(iris.mis$Sepal.Length),]

# Use the regression model (iris.mis.lm) to predict the
    Sepal.Length in tmp:
# Note: na.action = na.omit makes sure that rows that
    contain NA are treated correctly
predSepalLength = predict(iris.mis.lm, newdata = tmp,
    na.action = na.omit)

# Compute a relative error for the predictions (since we
    have the truth in iris)
for (i in names(predSepalLength)) {
  t = iris$Sepal.Length[as.integer(i)]
  print(100*(predSepalLength[i] - t)/t)
}
```

In practice, you will not have to implement imputation methods yourself. There are specialised packages for the task. In the next section we will use one.

Hot Deck Imputation: Predictive Mean Matching

Regression Imputation may lead to predictions that are not valid, for example, negative lengths in the iris data. Hot deck imputation overcomes this problem. In this family of methods, missing values are imputed by copying values from similar entries in the same dataset. For instance, in *Random Hot Deck Imputation*, a value is chosen randomly, and uniformly. Another Hot Deck method is *Predictive Mean Matching (PMM)*. It is similar to the regression method (from the previous section), but has two additional steps. Suppose we have, as above, the variables $x_1, x_2, \ldots x_k$, and x_1 has missing values.

1. Estimate the relationship between x_1 and $x_2, \ldots, x_k$ by linear regression.
2. Use the regression model to make predictions for *all* the values in x_1, for the missing ones and for the non-missing ones.
3. Suppose there is a missing value in the i-th row, i.e. x_{1i} is missing. An let us denote the corresponding predicted value from step 2 with $\hat{x}_{1i}$. *Regression imputation* would substitute the missing value by $\hat{x}_{1i}$. In contrast, *PMM* finds a set of rows with non-missing values, whose predicted values are close to the predicted value $\hat{x}_{1i}$.
4. From among those close cases, one case is randomly chosen and its *observed* value is used to substitute for the missing value x_{1i}.

PMM is implemented in the in the package *Mice* and can be invoked as follows:

```
temp = mice(iris.mis, meth="pmm")
iris.pmm = complete(temp)
iris.pmm

# Remove the categorical variable "Species" from the
    original dataset:
iris.num = subset(iris, select=-c(Species))
# Compute a table with relative errors:
err = (iris.pmm-iris.num)/iris.num
summary(err)
```

Chapter 4

Data visualisation

4.1 Introduction to ggplot

R has built-in functions for visualisation, and there exist a number of packages for the task. *ggplot* is one of the most popular ones, or maybe the most popular. It is very flexible and has *a logic* for the mapping between the data and its representation, allowing a *graphical construction* of plots. The author of the package is Hadley Wickham; consequently, ggplot assumes data in tidy format (see Section 3.1).

The goal here is to introduce you to the principles of ggplot using basic graphs. Once you grasp these principles, you should be able to defend yourself –with R's help system, with turorials, and with the many examples that can be found on the internet (due to the popularity of ggplot). Search engines are your friend here.

The section *The grammar of ggplot* draws from a paper by Wickham on the topic [9] and his main book on ggplot [8]. The section *ggplot by examples with basic graphs* follows a tutorial[1] published at *RPubs*. RPubs is a collection of tutorials provided by *RStudio*

You will need the following packages:

```
library("ggplot2")
library("reshape2")
library("ggthemes")
library("devtools")
library("ggExtra")
library("psych")
```

[1] Short Introduction to ggplot2, Chuan Tang
https://rpubs.com/karagawa/ggplot2

4.1.1 The underlying grammar of ggplot

The grammar of ggplot follows the insights from Leland Wilkison's Grammar of Graphics [12] (hence "gg"). This makes it powerful, because you are not limited to a set of pre-specified graphics, but you can create new graphics that are tailored for your problem.

Building a basic plot

Consider the following simple data set:

A	B	C	D
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

We would like to draw a scatterplot of A versus C. What exactly is a scatterplot? One way to describe it is that we are going to draw a point for each observation, and we will position the point horizontally according to the value of A, and vertically according to C. For this example, we will also map categorical variable D to the shape of the points. The first step in making this plot is to create a new dataset that reflects the mapping of x-position to A, y-position to C, and shape to D. x-position, y-position, and shape are examples of *aesthetics, things that we can perceive on the graphic*. We will also remove all other variables that do not appear in the plot. This is shown in the following table:

x	y	Shape
2	4	a
1	1	a
4	15	b
9	80	b

We can create many different types of plots using this same basic specification. For example, if we were to draw lines instead of points, we would get a line plot. If we used bars, we would get a bar plot. *Bars, lines, and points are all examples of geometric objects.*

The next thing we need to do is to convert these numbers measured in data units to numbers measured in physical units, things that the computer can display. To do that we need to know that we are going to use linear scales and a Cartesian coordinate system. We can then convert the data units to aesthetic units, which have meaning to the underlying drawing system. For example, to convert from a continuous data value to a horizontal pixel coordinate, we need a function like, for example, the following:

$$\text{round}\left(\frac{x - \min(x)}{\max(x) - \min(x)} \cdot \text{screen width}\right)$$

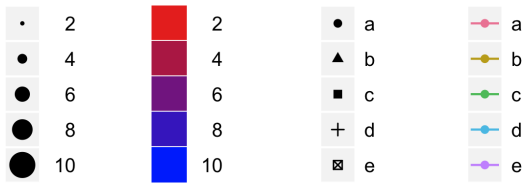


Figure 4.1: Examples of legends from four different scales. From left to right: continuous variable mapped to size and color, discrete variable mapped to shape and color.

For example, assuming the screen width to be 200 pixels, the x value 4 is converted to

$$\text{round}\left(\frac{4-1}{9-1}\cdot 200\right) = 75.$$

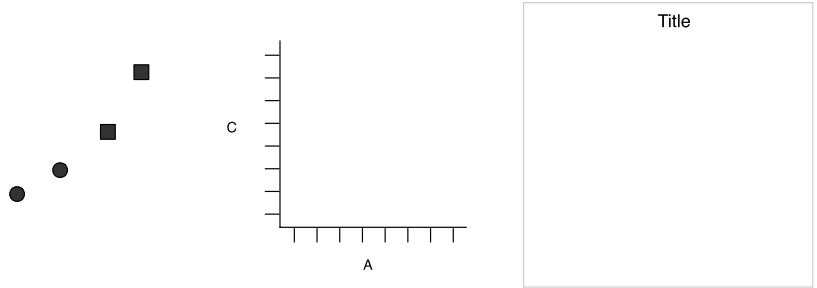
The results of the conversion are shown in the following table:

x	y	Shape
25	11	circle
0	0	circle
75	53	square
200	300	square

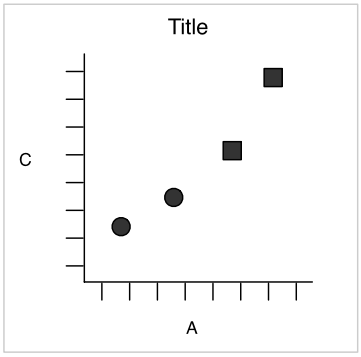
The procedure is similar for other aesthetics, such as shape: here *a* maps to a circle, and *b* to a square. *This transformation is the responsibility of scales* (see Figure 4.1). Finally, we need to render these data to create the graphical objects that are displayed on screen or paper. To create a complete plot we need to combine graphical objects from three sources:

1. The data, represented by the point geometry
2. The scales and coordinate system, which generates axes and legends so that we can read values from the graph
3. The plot annotations, such as the background and plot title.

These objects are shown in the following figure:



And here is the final graph, produced by combining the objects:



Components of the layered grammar

In the example above, we have seen some of the components that make up a plot:

- Data
- Aesthetic mappings (including scales)
- Geometric objects

Together these components form one layer. There can be an additional component in a layer, namely a statistical transformation. In the examples below you will encounter

- Bins in a histogram: dividing a continuous range into bins, and count the number of points in each
- Statistics necessary for a box plot
- Jitter values: adding a small random value to a variable

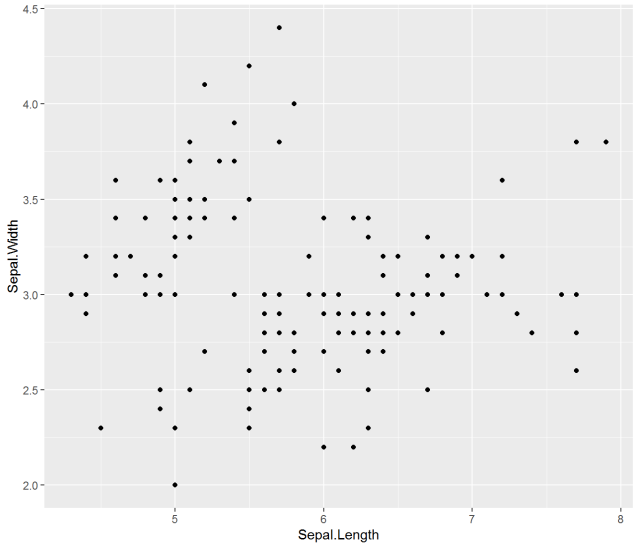
As we will see in the next section, *a plot may have multiple layers*, for example, when we overlay a scatterplot with a smoothed line.

4.1.2 ggplot by examples with basic graphs

In this section we explore ggplot by means of basic graphs (scatter plot, histogram, box plot). The envisioned approach is *learning by doing*. Feel free to experiment with the various parameters and parameter settings, and to try out your own ideas by modifying the given code.

Scatter plot

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_point()
```



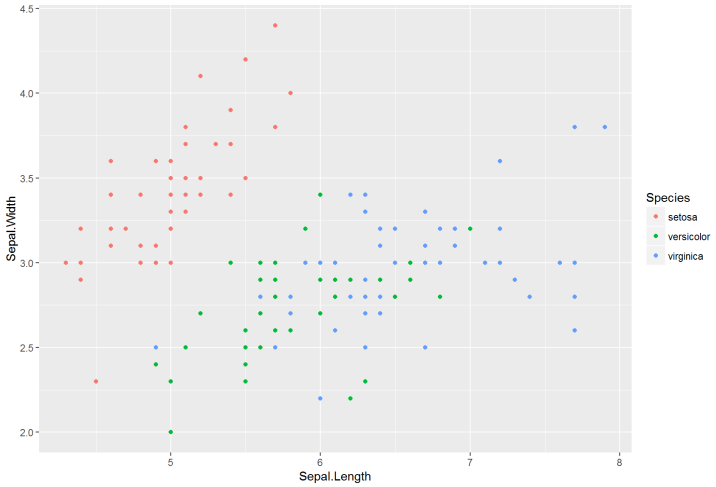
Apart from assigning variables to the x and y axis, you can assign a variable to distinguish groups. Here we chose the variable *species* for grouping. We can group by color:

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, colour
    = Species)) + geom_point()
```

In this example, the three components of the grammar (mentioned above) are:

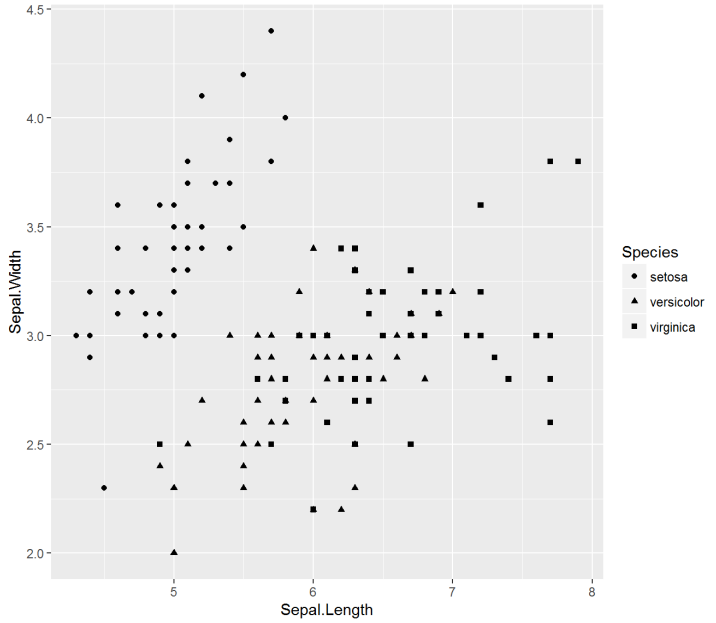
- Data: *iris*
- Aesthetic mappings: *aes(x = Sepal.Length, y = Sepal.Width, colour = Species)*, including the scale responsible for *colour = Species*
- Geometric objects: *geom_point()*

Here is the resulting graph:



We can also group by shape:

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, shape =
    Species)) + geom_point()
```

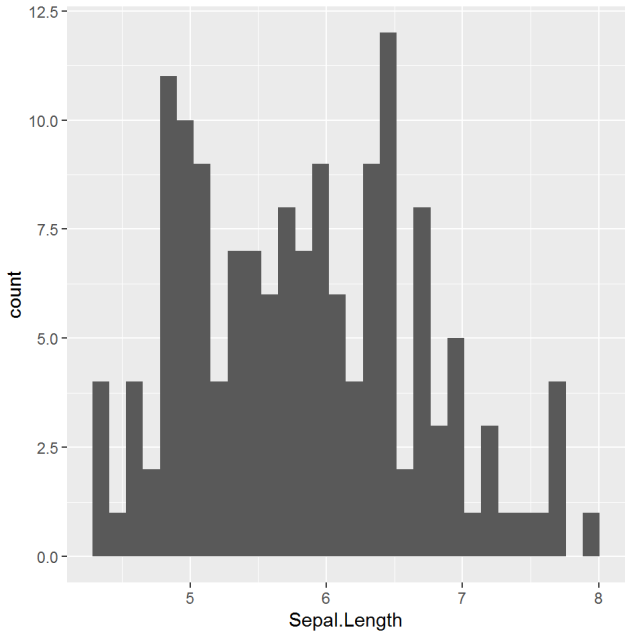


Exercise 4.1.1 *In this exercise you will use the built-in data set* mtcars. *You can learn about the dataset with R's help function:* help(mtcars). *Make a scatterplot as above. Assign the variable* mpg *to the x-axis,* wt *to the y-axis, and* gear *to the colour. This does not lead to the desired result. Why? (Compare the data type of* Species *and* gear*). How can you solve the problem?*

Histogram

Now, we get back to the iris dataset. We want to graph the distribution of *Sepal.Length* (over all Species).

```
ggplot(iris, aes(x = Sepal.Length)) + geom_histogram()
```



Exercise 4.1.2 *Make a density graph by simply adding a* geom_density() *layer, instead of* geom_histogram().

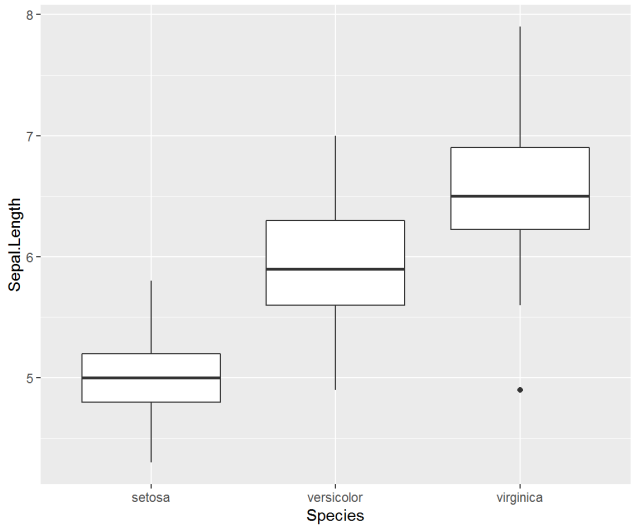
Exercise 4.1.3 *In the same way as with the scatter plots, add* colour= Species, *once to the histogram and once to the density graph.*

Exercise 4.1.4 *Replace* colour= Species *by* fill = Species *and see what happens.*

Boxplot

Boxplot is a standardized way of illustrating the distribution of data based on five important numbers (minimum, first quartile, median, third quartile, and maximum).

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) +
    geom_boxplot()
```



Exercise 4.1.5 *Add* + coord_flip() *to the boxplot. What happens?*

Exercise 4.1.6 *What is the difference between the first two commands in the code below? Does it make sense? Also try out the third command.*

```
1 ggplot(iris, aes(Species, Sepal.Length)) + geom_boxplot() +
      geom_point(position = "jitter")
2 ggplot(iris, aes(Species, Sepal.Length)) +
      geom_point(position = "jitter") + geom_boxplot()
3 ggplot(iris, aes(Species, Sepal.Length)) +
      geom_point(position = "jitter") + geom_boxplot(alpha =
      0.8)
```

Note: The parameter *position = "jitter"* is not essential to the exercise. It adds a bit of noise to the data and is used to unhide hidden data. You can leave it out to see its effect. It is a useful tool for data exploration, but you have to be careful when you use it for a general audience as it can lead to misinterpretation.

4.1.3 Aesthetics and more than one layer

There is flexibility in the ways to define aesthetics. This becomes particularly important, when you have more than one layer. In this section we approach this topics by means of two exercises.

Exercise 4.1.7 *Generate the following four plots. Are there any differences?*

```
1 ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, colour
      = Species)) + geom_point()
```

```
2 ggplot(iris) + geom_point(aes(x = Sepal.Length, y =
      Sepal.Width, colour = Species))
3 ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
      geom_point(aes(colour = Species))
4 ggplot(iris, aes(x = Sepal.Length)) + geom_point(aes(y =
      Sepal.Width, colour = Species))
```

Exercise 4.1.8 *Now we add a trend line to the scatter plot using the layer* geom_smooth(). *Run the following four variants, observe their differences, and try to understand the causes.*

```
1 ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, colour
      = Species)) + geom_point() + geom_smooth()
2 ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
      geom_point(aes(colour = Species)) + geom_smooth()
3 ggplot(iris) + geom_point(aes(x = Sepal.Length, y =
      Sepal.Width, colour = Species)) + geom_smooth()
4 ggplot(iris) + geom_point() + geom_smooth(aes(x =
      Sepal.Length, y = Sepal.Width, colour = Species))
```

Note: With *geom_smooth(method = lm)* you can add a regression line.

4.1.4 Faceting, multiple plots, saving plots

Faceting generates small multiples, each showing a different subset of the data. With small multiples, you can rapidly compare patterns in different parts of the data.

```
base = ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width))
    + geom_point()
g1 = base + facet_grid(Species ~ .)
g2 = base + facet_grid(. ~ Species)
```

Multiple plots can be arranged in a grid with the function *grid.arrange()* from the package *gridExtra*. It takes each plot you would like to include as a separate argument. Further, you can specify the number of columns in the grid and optionally give a title to the whole figure. The following example assumes that you have generate four plots *p1, ..., p4*:

```
library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol = 2, top = "Main title")
```

Saving plots: The following line of code saves a plot stored in the variable *g* under the file name *myplot.pdf* in the PDF format:

```
ggsave("myplot.pdf", g)
```

The function *ggsave()* recognizes

Exercise 4.1.9 *Produce four plots and arrange them in a grid with* grid.arrange(). *Experiment with the parameter* ncol.

4.1.5 And now?

We have introduced the principles of ggplot's grammar, based on basic plots. There are many things we have not covered here, for instance, legends, changing the label of axis, placing text in a plot, selecting fonts (and their size and orientation), or more Geoms. However, you should be equipped now with the basics so that you can look this elements up yourself; ggplot is very popular so that the internet is populated with examples. RStudio provides a nice cheat-sheet that may be usefull in this context.[2]

4.2 Special plots: Tufte in R with ggplot

Edward Tufte is an American statistician and professor statistics and computer science at Yale University and a pioneer in the field of data visualization. In the course there will be a lecture giving a tour on Tufte's classic book on statistical charts, graphs, and tables *The Visual Display of Quantitative Information.*

There is a wonderfull page by Lukasz Piwek, where many of the examples from Tufte's Visual Display are recreated in R:

http://motioninsocial.com/tufte/

The graphs have been created for Base Graphics, the *Lattice* package, and with *ggplot*. Select a plot (or more than one) that you like and work through the generating *ggplot* code. Run it, try to understand it, and modify parts of it. The examples include many elements mentioned above (e.g. legends, modifying axis, placing text, using themes). Do not work with the examples *Slopegraph* and *Sparklines*.

[2]The cheat sheet is available here:
https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Chapter 5

Exploring and Describing Data

5.1 Introduction

5.1.1 Statistics in a nutshell

The science of **statistics** deals with the collection, analysis, interpretation, and presentation of **data**. For example, a survey reveals the average time (in hours) that a group of students sleep per night. The collected data are the following,

$$5;\ 5.5;\ 6;\ 6;\ 6;\ 6.5;\ 6.5;\ 6.5;\ 6.5;\ 7;\ 7;\ 8;\ 8;\ 9\,,$$

and Figure 5.1 is a **dot plot** of the above data.

Figure 5.1: Dot plot representing number of sleeping hours per night.

Where do the data appear to cluster in the example above? How could the clustering be interpreted? If the same survey was carried out in a different class, would the results be the same? These (and many other) questions require **analyzing and interpreting the data**. **This is the domain of statistics**.

The ultimate goal of statistics is to gain useful information from data. In general, statistics provides methods for

- **Design**: Planning and carrying out research studies.
- **Description**: Summarizing and exploring data.
- **Inference**: Making predictions and generalizing about phenomena represented by the data.

We will focus here only on the middle point, so called **descriptive statistics**. The next paragraphs are intended to give a very brief overview of some fundamental concepts, which will then be analyzed in more details in the following Sections 5.2 and 5.3. For further reading, there are many good textbooks on basic statistics. One of them is [5].

5.1.2 Population and sample

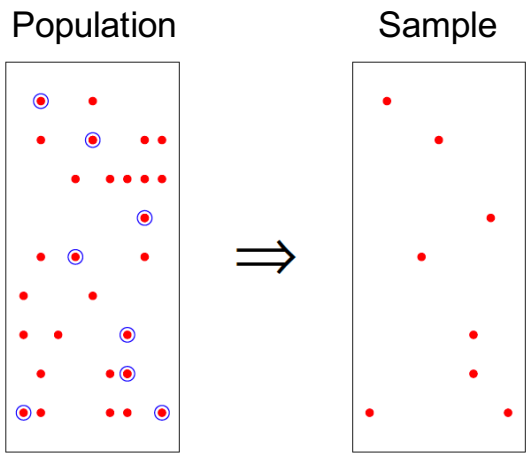


Figure 5.2: Population and sample.

Population and **sample** are two basic concepts of statistics. Population can be characterized as the entire set of individual persons or objects in which an investigator is interested. However, very often, for practical reasons only a subset of individuals of that population can be observed. Such a subset of individuals constitutes a sample.

In other words, the idea of sampling is to select a portion (or subset) of the larger population and study that subset to gain information about the entire population. **Data are the result of sampling from a population**. The population however always represents the target of an investigation.

Since it can take a lot of time (and money!) to examine an entire population, sampling is a very practical technique. For example, in presidential elections,

opinion poll samples of $1000 - 2000$ people are usually taken. The opinion poll is supposed to represent the views of the people in the entire country.

We can formalize these concepts with the following definitions.

Population is the collection of all individuals or items under consideration in a statistical study.

Sample is the subset of the original population from which information is collected.

From the sample data, we can calculate a **statistic**. A statistic is a number that represents a property of the sample. For example, if we consider one math class to be a sample of the population of all math classes, then the average number of points earned by students in that one math class at the end of the term is an example of a statistic. The statistic is an estimate of a **population parameter**. A parameter is a number that represents a property of the population. Since we considered all math classes to be the population, then the average number of points earned per student over all the math classes is an example of a parameter. One of the main concerns in the field of statistics is how accurately a statistic estimates a parameter. This depends on how well the sample represents the population. The sample must contain all the relevant information about the population in order to be a **representative sample**.

Consider as an example the outcomes of tossing a dice. The dice is rolled 100 times and the results are forming the sample data, which can be organized into the self-descriptive Table 5.1 or alternatively into the histogram of Fig. 5.3. The latter shows the possible outcomes (from 1 to 6) and their occurrences on the horizontal and vertical axis, respectively.

Roll outcome	Frequencies in the sample data
1	10
2	20
3	18
4	16
5	11
6	25

Table 5.1: Table representing 100 outcomes of tossing a dice.

Could one say that the most probable outcome of tossing the dice is 6? Obviously, it would be a completely wrong conclusion. The problem here is that **the sample (100 outcomes) is not a representative one**. In other words, its size is not sufficient to capture all the relevant information that would be virtually contained in a population comprising an infinite number of outcomes. Since tossing a dice an infinite number of times is practically impossible, it will suffice to increase the sample size, for example to 5000, as shown in Fig 5.4. The latter suggests, correctly, that all outcomes have the same probabilities.

The dice example gives a simple and clear idea of the possible pitfalls that can occur in sampling from a population.

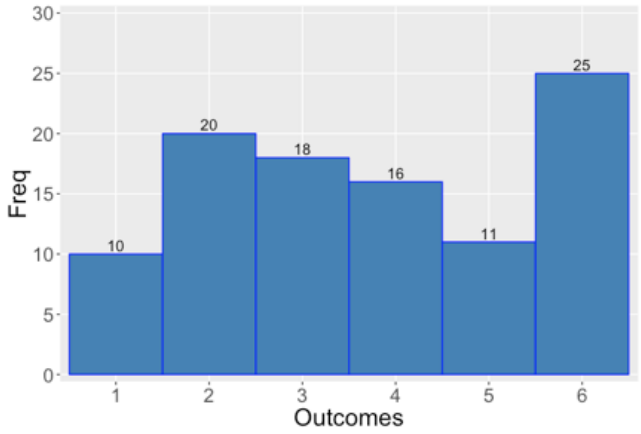


Figure 5.3: Histogram representing 100 outcomes of tossing a dice.

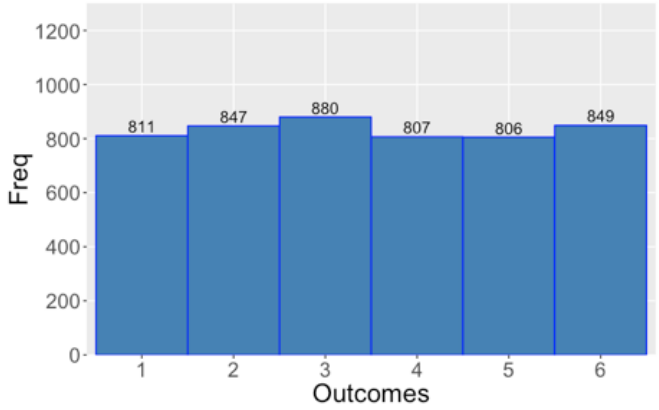


Figure 5.4: Histogram representing 5000 outcomes of tossing a dice.

5.1.3 Data

Data come in a wide range of formats. For example, a survey might ask questions about gender, race, or political affiliation, while other questions might be about age, income, or the distance to work covered each day. When data are collected, the information obtained from each member of a population or sample is recorded in the sequence in which it becomes available. This data sequence is random and unranked. Such data, before they are grouped or ranked, are called **raw data**.

There are two main types of data.

- **Categorical** (or **qualitative**) **data** record qualities or characteristics about the individual, such as eye color, gender, political party, or opinion

on some issue (using categories such as agree, disagree, or no opinion).

- **Numerical** (or **quantitative**) **data** record measurements or counts regarding each individual, which may include weight, age, height, or time to take an exam; counts may include number of pets, or the number of red lights you hit on your way to work.

The important difference between the two types of data is that with categorical data, any numbers involved do not have real numerical meaning (for example, using 1 for male and 2 for female), while all numerical data represents actual numbers for which math operations make sense.

However, it is possible that numbers associated with categorical variables might be ordered. Indeed, the categories into which a qualitative variable falls may or may not have a natural ordering. For example, occupational categories have no natural ordering. If the categories of a qualitative variable are unordered, then the qualitative variable is said to be defined on a **nominal scale**, the word nominal referring to the fact that the categories are merely names. If the categories can be put in order, the scale is called an **ordinal scale**. Based on what scale a qualitative variable is defined, the variable can be called as a **nominal variable** or an **ordinal variable**. Examples of ordinal variables are education (classified e.g. as low, high) and "strength of opinion" on some proposal (classified according to whether the individual favors the proposal, is indifferent towards it, or opposes it), and position at the end of race (first, second, etc...).

Quantitative data may be either **discrete** or **continuous**. All data that are the **result of counting** are called quantitative discrete data. These data take on only certain numerical values. If you count the number of phone calls you receive for each day of the week, you might get values such as zero, one, two, or three. All data that are the **result of measuring** are quantitative continuous data. Measuring angles in radians might result in such numbers as $\pi, \pi/3, \pi/2, \pi, 3\pi/4$, and so on.

Exercise 5.1.1 *Consider a group of students carrying backpacks with books to school. Are the numbers of books in the backpacks discrete or continuous data? And the weights of the backpacks?*

Based on the time over which they are collected, data can be classified as either **cross-section** or **time-series** data. Cross-section data contain information on different elements of a population or sample for the same period of time. The information on incomes of 100 families for 2009 is an example of cross-section data. Time-series data contain information on the same element for different periods of time. Information on Swiss exports for the years 1983 to 2009 is an example of time-series data.

As an example, Table 5.2 compares the number of part-time and full-time students enrolled at two colleges, called College One and College Two. These are clearly qualitative data, and can be easily visualized by means of tables and graphs.

College One				College Two		
	Number	Percent			Number	Percent
Full-time	9200	40.9%		Full-time	4059	28.6 %
Part-time	13296	59.1%		Part-time	10124	71.4 %
Total	22496	100%		Total.	14183	100%

Table 5.2: Number of students enrolled at two colleges

The table displays both counts (that is, **frequencies**) and percentages (or **relative frequencies**). The percent columns make the comparison between the same categories in the colleges easier. D**isplaying percentages along with the numbers is always helpful, but it is particularly important when comparing sets of data that do not have the same totals**, such as the total enrollments for both colleges in this example. Notice for instance how much larger the percentage of part-time students at College Two is compared to College One. This particular feature might go unnoticed if the table included only counts.

Tables are a good way of organizing and presenting data. However, **graphs** can often be more helpful in displaying relevant information about the data. Two graphs that are commonly used to display qualitative data are **pie charts** and **bar graphs**. Figures 5.5 and 5.6 are a pie chart and a bar graph, respectively, representing the data of Table 5.2.

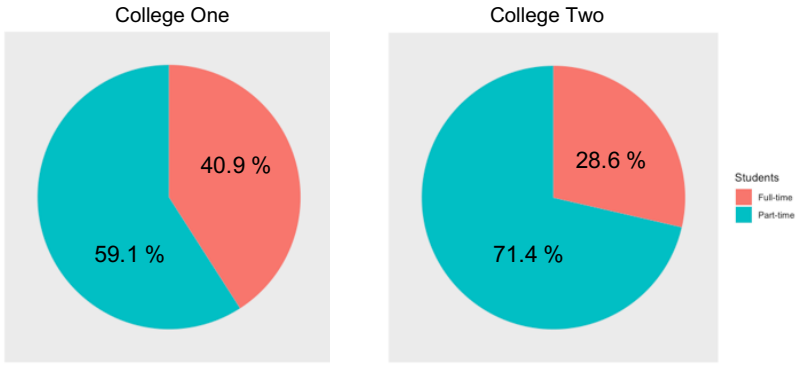


Figure 5.5: Pie chart representing the data of Table 5.2. Pie charts are a good graphical solution to display **percentages**.

It is a good idea to look at a variety of graphs to see which is the most helpful in displaying the data. We might make different choices of what we think is the best graph depending on the data and the context (and sometimes our personal taste).

So, for instance, percentages could add up to be more than 100%. This happens for example when individuals can belong simultaneously to more than one

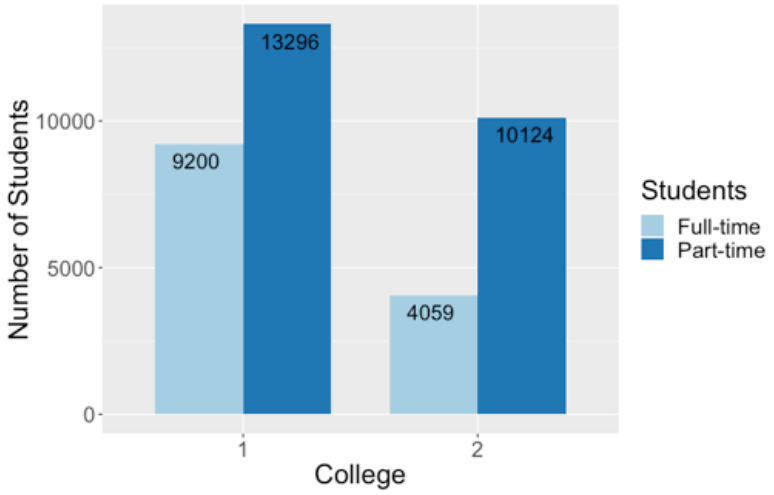


Figure 5.6: Bar graph representing the data of Table 5.2.

Category	**Percent**
Full-time students	40.9%
Students who intend to continue studies after college	48.6%
Students under age 25	61.0%
TOTAL	150.5%

Table 5.3: Categories of students at College One.

category. In Table 5.3, the percentages add to more than 100% because students can be in more than one category. In this case a bar graph is appropriate to compare the relative size of the categories (Figure 5.7), but a pie chart cannot be used. It also could not be used if the percentages added to less than 100%.

Let us now consider the ethnicity of students as presented in Table 5.4. The issue here is that a category "Other/Unknown" is missing. This category would contain people who did not feel they fit into any of the proposed ethnicity categories or declined to respond. Notice that as a consequence the frequencies do not add up to the total number of students. In this situation, a bar graph is a better option than a pie chart. Alternatively, the missing category "Other/Unknown" could be introduced *ad hoc* to take into account the 2338 students that do not belong to any of the suggested categories. Although with the additional category "Other/Unknown" a pie chart is possible (shown in Fig. 5.8), a bar graph, with the bars sorted from largest to smallest (called a **Pareto chart**), turns out to be very easy to read and interpret (see Fig. 5.9).

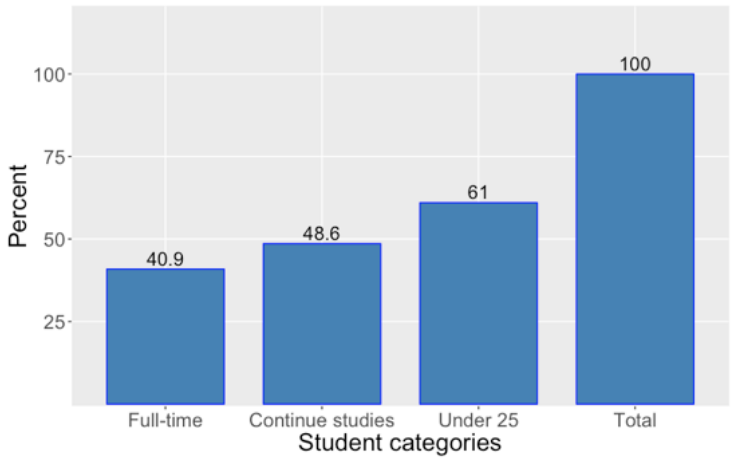


Figure 5.7: Bar graph representing the categories of students of Table 5.3.

	Frequency	**Percent**
Asian	8794	36.1%
Black	1412	5.8%
Filipino	1298	5.3%
Hispanic	4180	17.1%
Native American	146	0.6%
Pacific islander	236	1.0%
White	5978	24.5%
TOTAL	22044 out of 24382	90.4% out of 100%

Table 5.4: Ethnicity of students.

5.1.4 Sampling

Gathering information about an entire population often costs too much or is virtually impossible. Therefore, one has to resort to using a **sample of the population**. **A sample should have the same characteristics as the population that it is supposed to represent**.

- **Random sampling** techniques rely on the concept that all the members of a population initially have an equal chance of being selected for the sample. So, a **simple random sample** is selected assuming that any group of n individuals is equally likely to be chosen as any other group of the same size.

More advanced sampling methods include the stratified sample, the cluster sample, and the systematic sample.

- To choose a **stratified sample**, the population is divided into groups

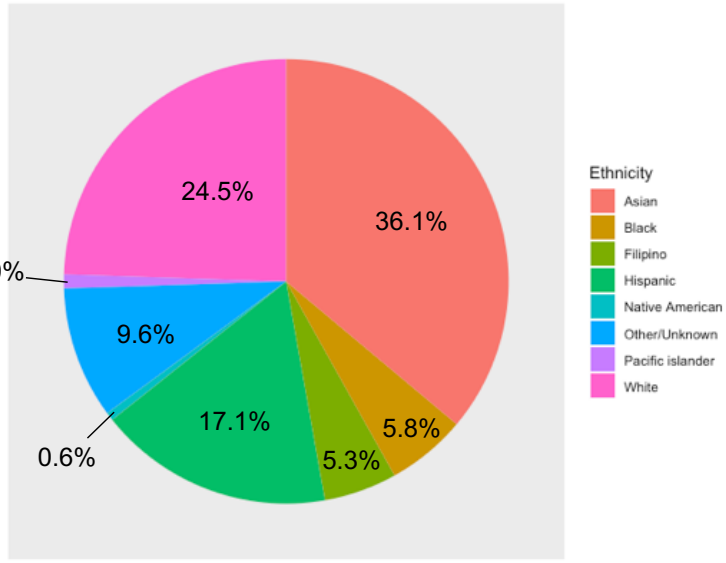


Figure 5.8: Pie chart showing the ethnicity categories of Table 5.4. The category "Other/Unknown" was added *ad hoc* to account for the students that did not belong to any of the proposed categories.

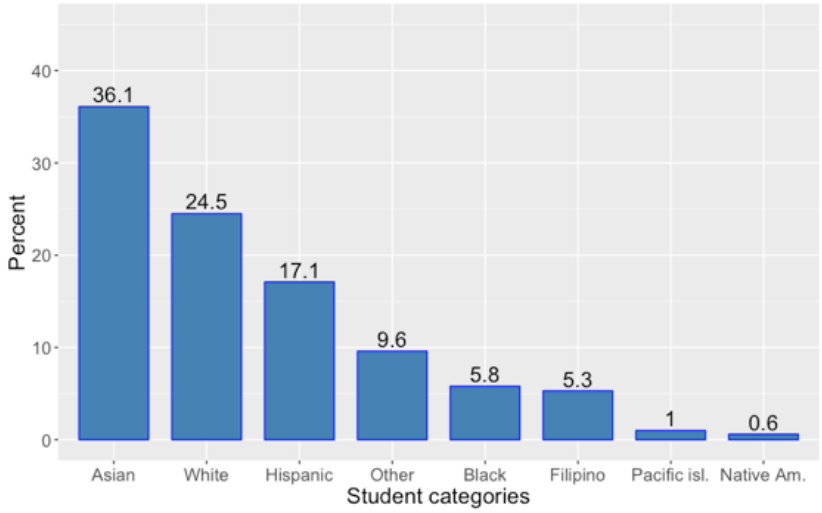


Figure 5.9: Pareto chart showing the ethnicity categories as in Figure 5.8.

called strata and then a proportionate number of members is selected from each stratum. For example, you could stratify (group) your college population by department and then choose a proportionate simple random

sample from each stratum (each department) to get a stratified random sample.

- To choose a **cluster sample**, the population is divided into clusters (groups). Some of the clusters are then randomly selected. All the members from these clusters are in the cluster sample. For example, if you randomly sample four departments from your college population, the four departments make up the cluster sample.

- To choose a **systematic sample**, a starting point is randomly chosen and then every n^{th} piece of data from a listing of the population is selected. Consider for example a phone survey. The phone book contains 20000 residence listings, out of which 400 names have to be chosen for the sample. The population can be numbered, from 1 to 20000, and a simple random sampling method can be applied to pick a number that represents the first name in the sample. Then every 50^{th} name after the initial one is selected until the sample includes a total of 400 names (note that one might have to go back to the beginning of the phone list).

In general, a sample will never be exactly representative of the population and there will always be some sampling errors. **As a rule, the larger the sample, the smaller the sampling error**.

A type of sampling that is non-random is **convenience sampling**. Convenience sampling involves using results that are readily available. For example, a computer software store conducts a marketing study by interviewing potential customers who happen to be in the store browsing through the available software. However, one must be careful because convenience sampling might be easier and faster than other methods but can hide dangerous pitfalls. The results of convenience sampling may be very good in some cases but highly biased (that is, favor certain outcomes) in others. In statistics, a sampling **bias** is created when a sample is collected from a population and some members of the population are not as likely to be chosen as others (remember, each member of the population should have an equally likely chance of being chosen). When a sampling bias happens, there can be incorrect conclusions drawn about the population that is being studied.

As an example, suppose Bob wants to know the opinions of people in his city regarding a proposed casino. Bob goes to the mall with his clipboard and asks people who walk by to give their opinions. What is wrong with that? Well, Bob is only going to get the opinions of a) people who shop at that mall; b) on that particular day; c) at that particular time; d) and who take the time to respond. That is too restrictive because those people dont represent a cross-section of the city. Similarly, Bob could put up an on-line survey and ask people to use it to vote. However, only those who know about the site, have Internet access, and want to respond will give him data. Typically, only those with strong opinions will go to such trouble. So, again, these individuals do not represent all the population in the city.

Randomness is of paramount importance in sampling methods. All the sampling methods mentioned above are conceived to minimize bias.

5.1.5 Frequency distributions

Let us now consider now a **quantitative variable**. Suppose twenty students were asked how many hours they worked per day. Their responses, in hours, are as follows: 5, 6, 3, 3, 2, 4, 7, 5, 2, 3, 5, 6, 5, 4, 4, 3, 5, 2, 5, 3. These outcomes can be organized and presented in the form of a **frequency table**, or **frequency distribution**, as shown in Table 5.5.

Working hours	**Frequency**	**Relative frequency**	**Cumulative rel. freq.**
2	3	3/20 ($\approx 15\%$)	$\approx 15\%$
3	5	5/20 ($\approx 25\%$)	$\approx 40\%$
4	3	3/20 ($\approx 15\%$)	$\approx 55\%$
5	6	6/20 ($\approx 30\%$)	$\approx 85\%$
6	2	2/20 ($\approx 10\%$)	$\approx 95\%$
7	1	1/20 ($\approx 5\%$)	$\approx 100\%$

Table 5.5: Frequency table showing from left to right data values, frequencies, relative frequencies and cumulative relative frequencies.

A **frequency** is the number of times a value of the data (working hours in this case) occurs. According to Table 5.5, there are three students who work two hours, five students who work three hours, and so on. The sum of the values in the frequency column, 20, represents the total number of students included in the sample.

A **relative frequency** is the ratio (fraction or proportion) of the number of times a value of the data occurs to the total number of outcomes. Relative frequencies can be written as fractions or percents. The sum of the values in the relative frequency column is 20/20, or 100%.

Cumulative relative frequency is the "accumulation" of the previous relative frequencies. To find the cumulative relative frequencies, add all the previous relative frequencies to the relative frequency for the current row, as shown in the rightmost column of Table 5.5. The fourth line of Table 5.5 for instance tells us that 85% of the students work five hours or less each day.

If quantitative data is discrete with only few possible values, as in our case with working hours, then the variable is best presented graphically by a bar graph, as shown in Figure 5.10.

If the quantitative variable under investigation can have a lot of different values, or it is a continuous variable, then the data must be grouped into classes (categories) before a frequency table can be constructed. The main steps in a process of grouping a quantitative variable into classes are the following.

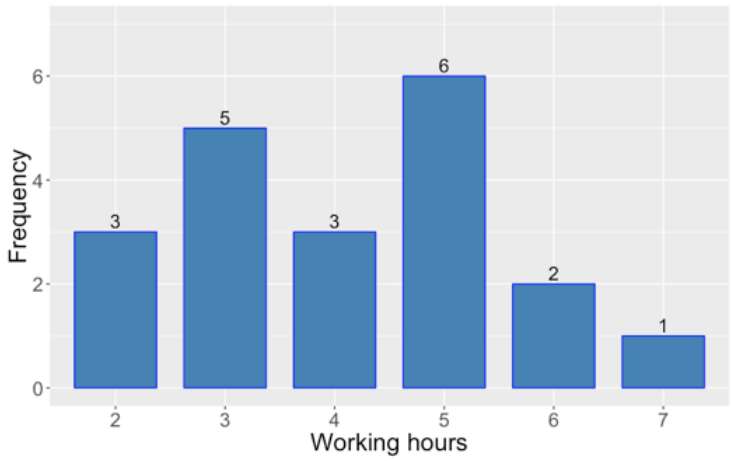


Figure 5.10: Bar chart showing the frequency distribution for the working hours of Table 5.5.

- Find the minimum and the maximum values of the variable in the data set.
- Choose intervals of equal length that cover the range between the minimum and the maximum without overlapping. These are called **class intervals**, and their end points are called **class limits**.
- Count the number of observations in the data that belongs to each class interval. The count in each class is the **class frequency**.
- Calculate the relative frequencies of each class by dividing the class frequency by the total number of observations in the data.

Frequencies or relative frequencies of each class interval can then be presented graphically in the form of a histogram or a bar chart. Let us consider for example the age of 102 people in the range 18 to 72 years. One can define eleven class intervals of 5 years each and group the data as shown in Table 5.6. The corresponding histogram is shown in Figure 5.11.

Qualitative data are usually presented graphically either as pie charts or bar graphs. As a rule of thumb, nominal data is best displayed by pie charts and ordinal data by bar graphs. Consider for example a survey about the blood types of 40 people. This is clearly qualitative nominal data. The outcomes could be:

$$O, O, A, B, A, O, A, A, A, O, B, O, B, O, O, A, O, O, A, A, A,$$
$$A, AB, A, B, A, A, O, O, A, O, O, A, A, A, O, A, O, O, AB\,.$$

Age class	Frequency	Relative frequency (%)	Cumulative rel. freq. (%)
18-22	6	5.9	5.9
23-27	10	9.8	15.7
28-32	14	13.7	29.4
33-37	11	10.8	40.2
38-42	19	18.6	58.8
43-47	8	7.8	66.7
48-52	12	11.8	78.4
53-57	12	11.8	90.2
58-62	4	3.9	94.1
63-67	2	2.0	96.1
68-72	4	3.9	100.0
TOTAL	102	100.0	

Table 5.6: Frequency distribution for the age of 102 people.

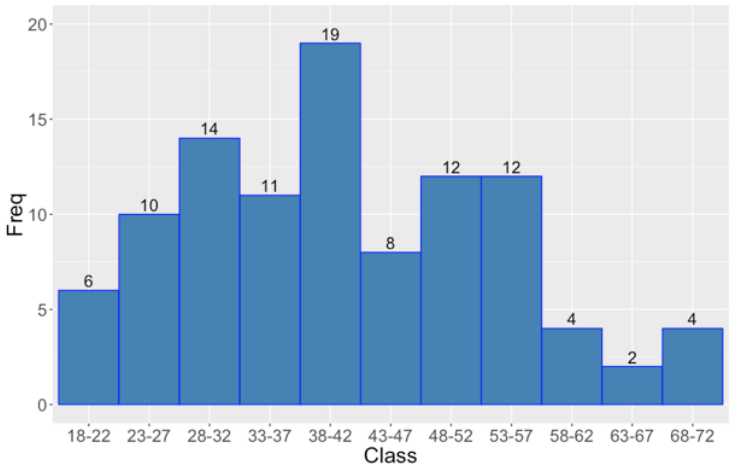


Figure 5.11: Histogram showing the frequency distribution for the age survey of Table 5.6.

These data can be easily summarized in a frequency table (Table 5.7) and graphically presented by a pie chart (Figure 5.12).

Frequency distributions for a variable apply both to a population and to samples from that population. The first type is called the population distribution of the variable, and the second type is called a sample distribution. In a sense, the sample distribution is a blurry photograph of the population distribution. As the sample size increases, the sample relative frequency in any class interval gets closer to the true population relative frequency. Thus, the photo-

Blood	Frequency	Rel. freq. (%)
O	16	40.0
A	18	45.0
B	4	10.0
AB	2	5.0
TOTAL	40	100.0

Table 5.7: Frequency distribution for the blood types of 40 people.

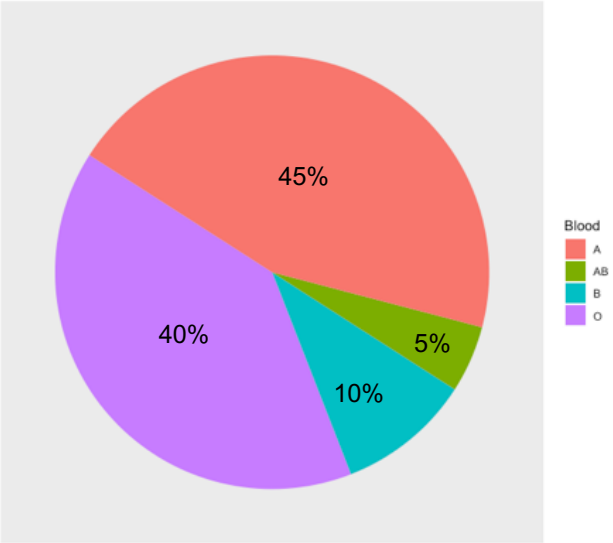


Figure 5.12: Pie chart for the blood types of Table 5.7.

graph gets clearer, and the sample distribution looks more like the population distribution. We have already come across this concept when discussing the outcomes of tossing a dice in Section 5.1.2.

When a variable is **continuous**, one can choose class intervals in the frequency distribution as narrow as desired. Now, as the sample size increases indefinitely and the number of class intervals simultaneously increases, with their width narrowing, the shape of the sample histogram gradually approaches a smooth curve. We use such curves to represent population distributions. Figure 5.13 shows two samples histograms, based on sample sizes of 100 and 10000, and a smooth curve (a "**density plot**") approximating the population distribution. The shape of the distribution can reveal important information.

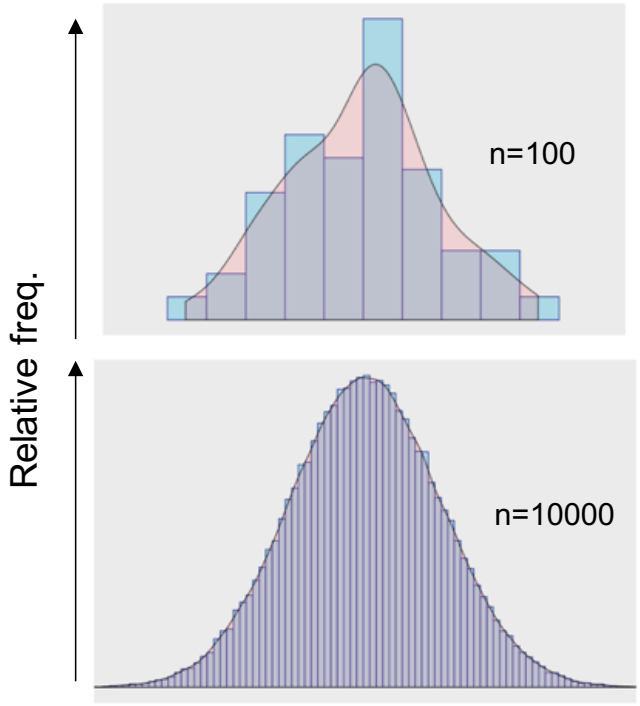


Figure 5.13: Sample frequency distributions based on two different sample sizes, $n = 100$ (top) and $n = 10000$ (bottom). The smooth curve (the **density** plot) approximates the corresponding population distribution.

5.2 Organizing and graphing data

In this chapter we will see how to organize and display data using tables and graphs. Most of the concepts and methods described below should be already familiar after reading Section 5.1. This Section is intended to provide a more detailed insight and robust understanding of how data can be organized and presented before any quantitative descriptive measure is calculated.

5.2.1 Raw data

Data recorded in the sequence in which they are collected and before they are processed or ranked are called **raw data**.

Suppose we collect information on the ages (in years) of 50 students selected from a university. The data values, in the order they are collected, are recorded in Table 5.8. For instance, the first student's age is 21, the second student's age is 19 and so forth. The data in Table 5.8 are **quantitative raw data**.

Suppose now we ask the same 50 students about their student status. The responses of the students are recorded in Table 5.9. In this table, F, SO, J,

Age of 50 students									
21	19	24	25	29	34	26	27	37	33
18	20	19	22	19	19	25	22	25	23
25	19	31	19	23	18	23	19	23	26
22	28	21	20	22	22	21	20	19	21
25	23	18	37	27	23	21	25	21	24

Table 5.8: Raw quantitative data: age of 50 students.

and SE are the abbreviations for freshman, sophomore, junior, and senior, respectively. This is an example of **qualitative (or categorical) raw data**.

Status of 50 students									
J	F	SO	SE	J	J	SE	J	J	J
F	F	J	F	F	F	SE	SO	SE	J
J	F	SE	SO	SO	F	J	F	SE	SE
SO	SE	J	SO	SO	J	J	SO	F	SO
SE	SE	F	SE	J	SO	F	J	SO	SO

Table 5.9: Raw quantitative data: age of 50 students.

The data presented in Tables 5.8 and 5.9 are also called **ungrouped data**. An ungrouped data set contains information on each member of a sample or population individually.

5.2.2 Analysis of qualitative data

In this section we shall discuss how to organize and display qualitative (or categorical) data. It should be clear after reading Section 5.1 that data will be organized into tables, and displayed using graphs.

Type of employment (category)	**Number of students (frequency)**
Private companies/business	44
Federal government	16
State/local government	23
Own business	17
TOTAL	100

Table 5.10: Frequency table for the type of career students intend to pursue.

A sample of 100 students enrolled at a US university were asked what they

intended to do after graduation. Among them, 44 said they wanted to work for private companies/businesses, 16 said they wanted to work for the federal government, 23 wanted to work for state or local governments, and 17 intended to start their own businesses. Table 5.10 lists the types of employment and the number of students who intend to engage in each type of employment. In this table, the variable is the type of employment, which is obviously a qualitative variable. The categories (representing the type of employment) listed in the first column are mutually exclusive. In other words, each of the 100 students belongs to one and only one of these categories. The number of students who belong to a certain category is called the frequency of that category. A frequency distribution exhibits how the frequencies are distributed over various categories. As already discussed in Section 5.1.5, Table 5.10 is called a **frequency distribution table** or simply a frequency table. **A frequency distribution for qualitative data lists all categories and the number of elements that belong to each of the categories**.

The **relative frequency** of a category is obtained by dividing the frequency of that category by the sum of all frequencies,

$$\text{Relative frequency of a category} = \frac{\text{Frequency of that category}}{\text{Sum of all frequencies}}\,, \tag{5.1}$$

while the **percentage** for a category is obtained by multiplying the relative frequency of that category by 100,

$$\text{Percentage} = \text{Relative frequency} \cdot 100\,. \tag{5.2}$$

Exercise 5.2.1 *A sample of 30 employees from large companies was selected and they were asked how stressful their jobs were. The responses of these employees are recorded in Table 5.11, where V represents very stressful, S means somewhat stressful, and N stands for not stressful at all. Construct a frequency distribution table for these data. Calculate also relative frequency and percentage distributions. Finally, check that the sums of frequencies, relative frequencies and percentages are correct.*

Amount of stress									
S	N	S	V	V	N	V	S	S	V
S	S	V	S	N	V	N	S	S	V
S	S	V	N	S	V	V	S	N	S

Table 5.11: Amount of stress at work, V = very stressful, S = somewhat stressful, N = not stressful.

A graphic display like a **bar graph** and a **pie chart** can reveal at a glance the main characteristics of a data set. We have already seen several of them in Section 5.1. We will summarize here their main features.

To construct a **bar graph** (also called a bar chart) one should follow the following guidelines.

- mark the various categories on the horizontal axis using intervals of the same width;
- mark the frequencies on the vertical axis;
- draw one bar for each category such that the height of the bar represents the frequency of the corresponding category (leaving a small gap between adjacent bars).

The following R code generates a bar graph for the frequency distribution of Table 5.10. The bar graph is shown in Fig. 5.14.

```
> df <- data.frame(Employment_type =
    c("Private","Federal","State","Own"),
     Freq = c(44,16,23,17))
> ggplot(data=df, aes(x=Employment_type, y=Freq)) +
    geom_bar(stat="identity", width=0.75, color="blue",
    fill="steelblue") + theme(text = element_text(size=25))
    + geom_text(aes(label=Freq), vjust=-0.3, size=7) +
    coord_cartesian(ylim = c(0,50))
```

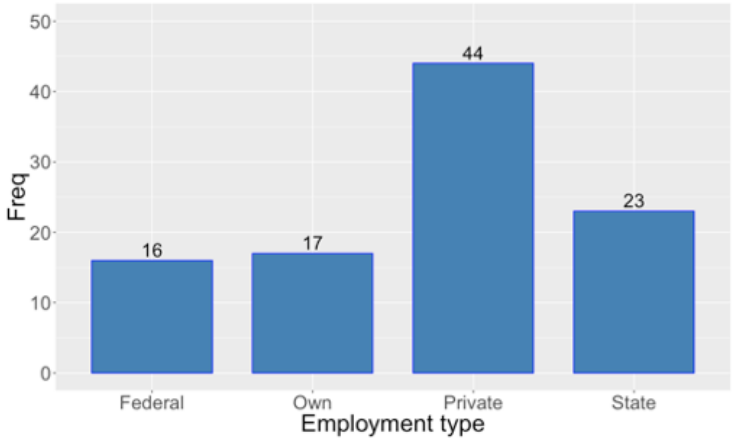


Figure 5.14: Bar graph representing the frequency distribution of Table 5.10.

We observe that, i) the bar graphs for relative frequency and percentage distributions can be drawn simply by marking the relative frequencies or percentages, instead of the frequencies, on the vertical axis; ii) a bar graph can also be constructed by marking the categories on the vertical axis and the frequencies on the horizontal axis (using coord_flip()).

A **pie chart** is more commonly used to display percentages, although it can be used to display frequencies or relative frequencies as well. To construct a pie chart one should follow these simple guidelines.

- draw a circle (i.e, the "pie") representing the total sample or population;

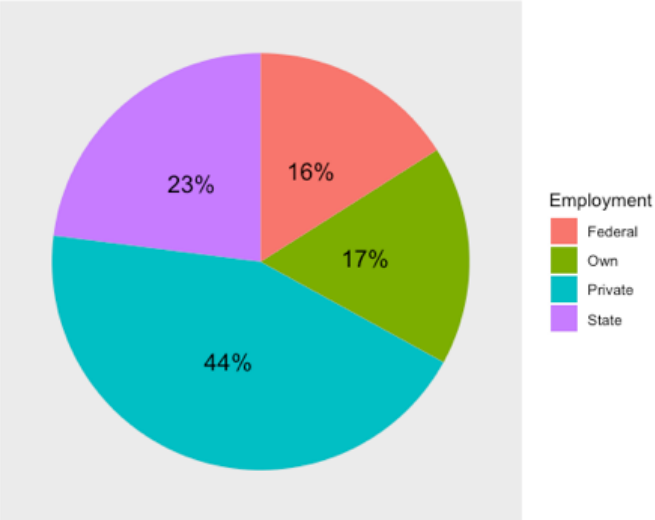


Figure 5.15: Pie chart representing the percentage distribution of Table 5.10.

- divide the pie into different portions representing the different categories;
- multiply 360 by the relative frequency of each category to obtain the degree measure of the angle for the corresponding category.

The following is the R code to generate a pie chart for the frequency Table 5.10, as shown in Figure 5.15.

```
> bp <- ggplot(df, aes(x="", y=Freq, fill=Employment_type))
    + geom_bar(width = 1, stat = "identity")
> bp + coord_polar("y", start=0, direction=-1) +
    theme(axis.title.x = element_blank(), axis.title.y =
    element_blank(), panel.border = element_blank(),
    panel.grid=element_blank(), axis.ticks =
    element_blank(), plot.title=element_text(size=14,
    face="bold")) + theme(axis.text.x=element_blank())
```

Exercise 5.2.2 *Consider again exercise 5.2.1. Construct a bar graph and a pie chart for the frequency distributions derived from the data of Table 5.11.*

Exercise 5.2.3 *In exit polls taken during the 2008 American presidential election, voters were asked to provide their education levels. Table 5.12 below summarizes their responses. Draw a bar graph and a pie chart to display these data.*

Education level	**Percentage of responses**
Not a high school graduate	4
High school graduate	20
Some college education	31
College graduate	28
Post graduate education	17

Table 5.12: Education level of voters for the 2008 presidential elections. Source: New York Times, November 5, 2008.

Exercise 5.2.4 *The data of Table 5.13 show the method of payment by 16 customers in a supermarket checkout line. Construct a frequency distribution table. Calculate the relative frequencies and percentages for all categories. Draw a bar graph for the frequency distribution and a pie chart for the percentage distribution.*

Payment methods							
C	CK	CK	C	CC	D	O	C
CK	CC	D	CC	C	CK	CK	CC

Table 5.13: Payment methods at a supermarket, C = cash, CK = check, CC = credit card, D = debit card, O = other.

5.2.3 Analysis of quantitative data

In the previous section we have seen how to group and display qualitative data. We shall see in this section how to group and display **quantitative data**.

Table 5.14 shows the weekly earnings of 100 employees of a large company. The first column lists the classes, which represent the (quantitative) variable "weekly earnings". For quantitative data, an interval that includes all the values that fall within two numbers, i.e., the lower and upper limits, is called a **class**. Note that the classes always represent a variable and are non-overlapping. The second column in the table lists the number of employees who have earnings within each class. The numbers listed in the second column are called the **frequencies** (denoted by "f").

For quantitative data, the frequency of a class represents the number of values in the data set that fall in that class. Table 5.14 is a **frequency distribution table for quantitative data**. Data presented in the form of a frequency distribution are called **grouped data**.

The **class boundary** is given by the midpoint of the upper limit of one class and the lower limit of the next class. So, for example, the upper boundary of the first class in Table 5.14, or equivalently the lower boundary of the second

Weekly earnings (classes)	**Number of employees (frequency f)**
801 - 1000 $	9
1001 - 1200 $	22
1201 - 1400 $	39
1401 - 1600 $	15
1601 - 1800 $	9
1801 - 2000 $	6

Table 5.14: Weekly earnings of 100 employees of a company.

class, is given by $(1000 + 1001)/2 = 1000.5$. Class boundaries are also called **real class limits**.

The difference between the two boundaries of a class gives the **class width**, also called the **class size**. One can easily verify that the size of the classes in Table 5.14 is 200 $.

The **class midpoint** or **mark** is obtained by dividing the sum of the two limits (or the two boundaries) of a class by 2. The midpoint of the first class in Table 5.14 is for instance 900.5, while the midpoint of the second class is 1100.5 and so forth.

A few considerations should be made regarding the construction of frequency distribution tables.

- The number of classes for a frequency distribution table is an arbitrary choice made by the data organizer and depends essentially on the number of observations in the data set. It is preferable to have more classes as the size of a data set increases.

- It is usually preferable to define classes with the same width. In this case, one should first find the difference between the largest and the smallest values in the data. Then, the approximate width of a class is obtained by dividing this difference by the number of desired classes. Usually this approximate class width is rounded to a convenient number, which is then used as the class width. Note that rounding this number may slightly change the number of classes initially intended.

- Any convenient number that is equal to or less than the smallest value in the data set can be used as the lower limit of the first class.

Let us consider another example. The following raw data give the total number of iPods sold by a mail order company on each of 30 consecutive days.

For constructing a frequency distribution table one should first identify the minimum and maximum values in the data set, that is, 6 and 29, respectively. Suppose one decides to group these data using five classes of equal width. Then, $(29 - 6)/5 = 4.6 \approx 5$ will be the class size. The lower limit of the first class can be taken as 5 or any number less than 5. Suppose one takes 5 as the lower limit

Sold iPods									
8	25	11	15	29	22	10	6	17	21
22	13	26	16	18	12	9	26	20	16
23	14	19	23	20	16	27	16	21	14

Table 5.15: Raw quantitative data: iPods sold daily by a mail order company.

of the first class. Then the classes will be $5-10, 11-15, 16-20, 21-25$ and $26-30$. It should now be straightforward to construct the frequency distribution Table 5.16.

iPods sold	f
5 - 10	4
11 - 15	6
16 - 20	9
21 - 25	7
26 - 30	4

Table 5.16: Frequency distribution table for the raw data of Table 5.15.

Relative frequencies and percentage distributions are calculated in the same way as defined for qualitative data in Section 5.2.2.

An example of R code to generate a frequency distribution table from the raw data of Table 5.15 is shown here. The class intervals and the associated frequencies are generated using the built-in function cut().

```
#### Create data frame with raw data
df_raw <- data.frame(Sold = c(8, 25, 11, [...], 16, 21, 14))

#### Define class boundaries and width
> low_val <- 5
> high_val <- 30
> step_val <- 5
> x_breaks <- seq(low_val, high_val, step_val); x_breaks
[1]  5 10 15 20 25 30

#### Generate classes and frequencies
> x <- cut(df_raw$Sold, breaks = x_breaks)

#### Create data frame
> df <- data.frame(table(x));

#### Generate relative frequencies and percentages
> df$Rel_freq <- df$Freq/sum(df$Freq)
> df$Percent <- df$Rel_freq*100
```

```
> df
         x Freq  Rel_freq  Percent
1  (5,10]    4 0.1333333 13.33333
2 (10,15]    6 0.2000000 20.00000
3 (15,20]    9 0.3000000 30.00000
4 (20,25]    7 0.2333333 23.33333
5 (25,30]    4 0.1333333 13.33333
```

5.2.4 Graphical representation of quantitative data

Grouped (quantitative) data can be displayed using a **histogram** or a **polygon**. One can also draw a pie chart to display the percentage distribution for a quantitative data set.

Histograms are constructed much in the same way as bar charts. Indeed, one should,

- mark classes on the horizontal axis and frequencies (or relative frequencies or percentages) on the vertical axis;
- draw a bar for each class so that its height represents the frequency of that class;
- remember to draw bars **adjacent to each other with no gap between them**.

A histogram is called a **frequency histogram**, a **relative frequency histogram**, or a **percentage histogram** depending on whether frequencies, relative frequencies, or percentages are marked on the vertical axis. A frequency histogram for the data of Table 5.16, generated using the following R code, is shown in Figure 5.16.

```
ggplot(data=df, aes(x=x,y=Freq)) +
    geom_bar(stat="identity", width=1, color="blue",
    fill="steelblue") + theme(text = element_text(size=25))
    + xlab("iPods sold") + geom_text(aes(label=Freq),
    vjust=-0.3, size=7.5) + coord_cartesian(ylim = c(0,10))
```

A **polygon** is another graphic tool that can be used to display quantitative data. To draw a frequency polygon, one should,

- mark a dot above the midpoint of each class at a height equal to the frequency of that class. This is the same as marking the midpoint at the top of each bar in a histogram;
- mark two more classes, one at each end, and mark their midpoints. Note that these two classes have zero frequencies;
- join the adjacent dots with straight lines. The resulting line graph is called a frequency polygon or simply a polygon.

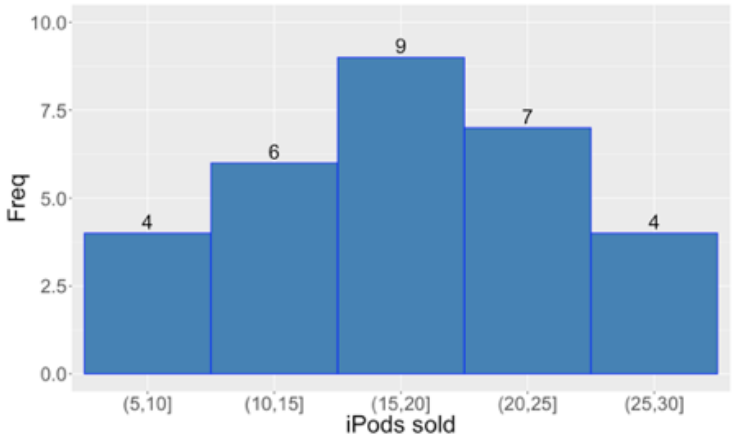


Figure 5.16: Frequency histogram for the data of Table 5.16.

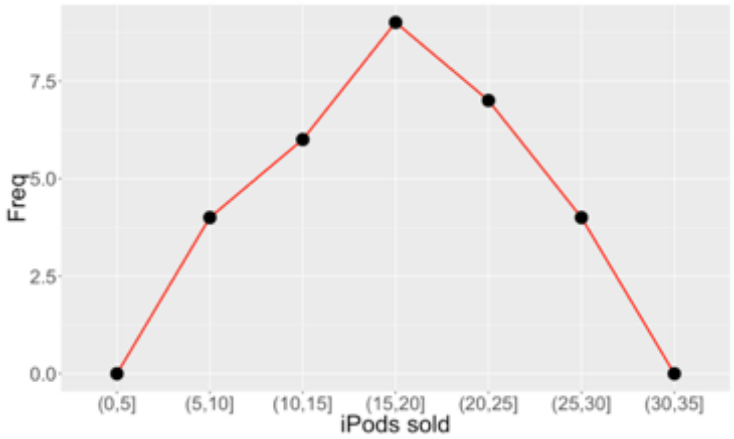


Figure 5.17: Frequency polygon for the data of Table 5.16.

Needless to say, a polygon with relative frequencies marked on the vertical axis is called a relative frequency polygon. Similarly, a polygon with percentages marked on the vertical axis is called a percentage polygon. A frequency polygon for Table 5.16 is shown in Figure 5.17. It was generated using the following R code. One might notice that two classes, with zero frequencies, had to be added at the two ends of the data frame.

```
> ggplot(data=df, aes(x=x, y=Freq, group=1)) +
    geom_line(aes(size=0.5), color="red") +
    geom_point(aes(size=1)) + theme(text =
    element_text(size=25)) + xlab("iPods sold") +
    theme(legend.position="none")
```

Exercise 5.2.5 *Construct a histogram and a polygon for the data of Table 5.14.*

Exercise 5.2.6 *Since 1996, Slate.com has published every year a list of the largest American charitable contributions by individuals. Table 5.17 gives the names of 22 persons on that list and the money they donated in 2008. Construct a frequency distribution table, including relative frequencies and percentages for all classes. Generate a histogram.*

Donor	**Donation** ($ millions)
Harold Alfond	360
Donald B. and Dorothy L. Stabler	334.2
David G. and Suzanne D. Booth	300
Frank C. Doble	272
Robert L. and Catherine H. McDevitt	250
Michael R. Bloomberg	235
Dorothy Clarke Patterson	225
Richard W. Weiland	174.3
Helen L. Kimmel	156.5
Jeffrey S. Skoll	144.1
H. F. (Gerry) and Marguerite B. Lenfest	139.9
David Rockefeller	137.8
Stephen A. Schwarzman	105
David H. Koch	100
Gerhard R. Andlinger	100
Eli and Edythe L. Broad	100
Philip H. and Penelope Knight	100
Kenneth G. and Elaine A. Langone	100
Fritz J. and Dolores H. Russ	94.8
Frank Sr. and Jane Batten	93
Jesse H. and Beulah C. Cox	83.5
Henry R. and Marie-Jose Kravis	75

Table 5.17: Largest American charitable donation by individuals for the year 2008. Source: Slate.com, January 26, 2009.

If the observations in a data set assume only a few distinct (integer) values, it may be appropriate to consider a particular type of frequency distribution table based on **single-valued** classes, that is, classes identified by single values instead of intervals. This approach is especially useful in cases of discrete data with only a few possible values.

Let us consider the following case as an example. The administration in a large city wanted to know the distribution of vehicles owned by households in that city. A sample of 40 randomly selected households from this city produced

Vehicles owned	Number of households (f)
0	2
1	18
2	11
3	4
4	3
5	2

Table 5.18: Frequency table for the number of vehicles per household.

the data of Table 5.18 on the number of vehicles owned. The observations in this data set assume only six distinct values: 0, 1, 2, 3, 4, and 5. Each of these six values is used as a class in the frequency distribution. In this particular case a bar graph might be a good solution for a graphical representation of the data (see Figure 5.18 and the R code below).

```
ggplot(data=df, aes(x = Vehicles, y = Freq)) +
    geom_bar(stat="identity", width=0.75, color="blue",
    fill="steelblue") + theme(text = element_text(size=25))
    + geom_text(aes(label=Freq), vjust=-0.3, size=7) +
    coord_cartesian(ylim = c(0,20)) +
    scale_x_discrete(limits = c(0,1,2,3,4,5)) +
    xlab("Vehicles owned")
```

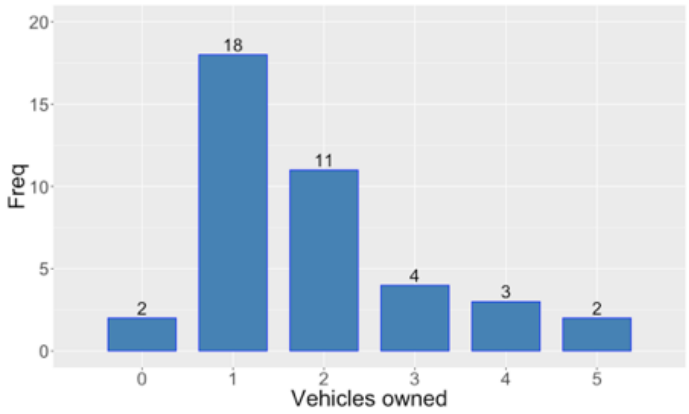


Figure 5.18: Bar graph representing single-valued classes.

As already briefly mentioned in Section 5.1.5, histograms can assume **many different shapes**. The most common of these shapes are i) **symmetric** (Figure 5.19), ii) **skewed** (Figure 5.20), iii) **uniform** or rectangular (Figure 5.21). The skewness of a histogram can be quantified by the R function skewness().

The latter requires installing and loading the library "e1071", as briefly discussed in Section 5.3.1.

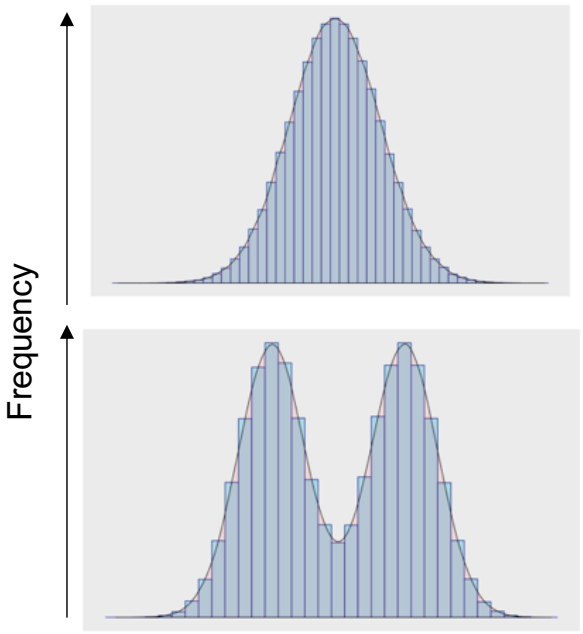


Figure 5.19: Examples of symmetric histograms.

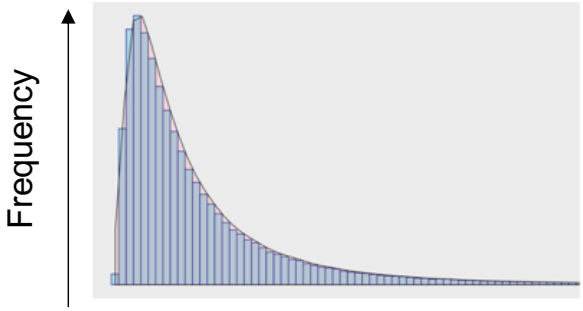


Figure 5.20: Skewed histogram with long tail to the right.

R provides functions to generate both histograms and density plots.

```
ggplot(df, aes(x=x)) +
  geom_histogram(aes(y=..density..), binwidth=1,
      color="darkblue", fill="lightblue") +
  geom_density(alpha=.1, fill="red")
```

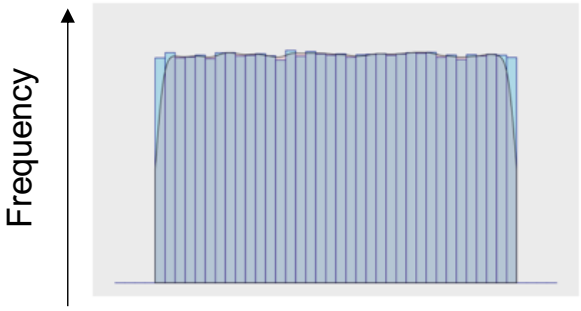


Figure 5.21: Histogram with uniform distribution showing the same frequency for each class.

Sometimes it might be necessary to specify customized bin widths and break points (i.e., where the bins are located). This was done for instance to generate Fig. 5.21.

```
ggplot(df, aes(x=x)) +
  geom_histogram(aes(y=..density..), binwidth=0.25,
      breaks=seq(0, 11, by=0.25), color="darkblue",
      fill="lightblue") +
  geom_density(alpha=.1, fill="red")
```

5.2.5 Cumulative frequency distributions

Consider again the data set of Tables 5.15 and 5.16 about the total number of iPods sold by a company. Suppose we want to know on how many days the company sold 20 or fewer iPods. Such a question can be answered by using a **cumulative frequency distribution**.

Each class in a cumulative frequency distribution table gives the total number of values that fall below a certain value.

A cumulative frequency distribution can be constructed for quantitative data only.

Table 5.19 yields the cumulative frequency distribution for the raw data of Table 5.16. This tells us that 20 or fewer iPods were sold on 19 days.

The **cumulative relative frequencies** are obtained by dividing the cumulative frequencies by the total number of observations in the data set. The **cumulative percentages** are obtained by multiplying the cumulative relative frequencies by 100. The following is the R code used to generate the cumulative frequencies of Table 5.19.

```
> c <- cumsum(df$Freq)
> n <- sum(df$Freq)
> crf <- c/n
> cper <- crf*100
```

iPods sold	Cumulative frequency
0 - 5	0
5 - 10	4
11 - 15	10
16 - 20	19
21 - 25	26
26 - 30	30

Table 5.19: Cumulative frequency distribution table for the raw data of Table 5.15.

```
> df$Cumul <- c
> df$Rel_cumul <- crf
> df$Per_cumul <- cper
> df
        x Freq Cumul Rel_cumul Per_cumul
1   (0,5]    0     0 0.0000000   0.00000
2  (5,10]    4     4 0.1333333  13.33333
3 (10,15]    6    10 0.3333333  33.33333
4 (15,20]    9    19 0.6333333  63.33333
5 (20,25]    7    26 0.8666667  86.66667
6 (25,30]    4    30 1.0000000 100.00000
```

A cumulative frequency distribution can be graphically represented by a curve called a **line graph**. Figure 5.22 is a line graph for the cumulative frequency distribution of Table 5.19. The dots are marked at the upper boundaries of various classes at heights equal to the corresponding cumulative frequencies. Note that the graph starts at the lower boundary of the first class (with value zero) and ends at the upper boundary of the last class.

Here is the R code used to generate the line graph of Figure 5.22.

```
> x_breaks
[1]  5 10 15 20 25 30
> df_new <- data.frame(Class_boundaries = x_breaks,
    Cumul_freq = c)
> ggplot(data=df_new, aes(x=Class_boundaries, y=Cumul_freq,
    group=1)) + geom_line(color="red",aes(size=0.25)) +
    geom_point(aes(size=0.5)) + theme(text =
    element_text(size=25)) + xlab("iPods sold") +
    ylab("Cumul. freq.") + coord_cartesian(xlim = c(0,35))
    + scale_x_discrete(limits=c(0,5,10,15,17,20,25,30,35))
    +scale_y_discrete(limits=c(5,10,13,15,20,25,30)) +
    theme(legend.position="none")
```

One advantage of a line graph is that it can be used to approximate the cumulative frequency for any interval. For example, we can use Figure 5.22 to find

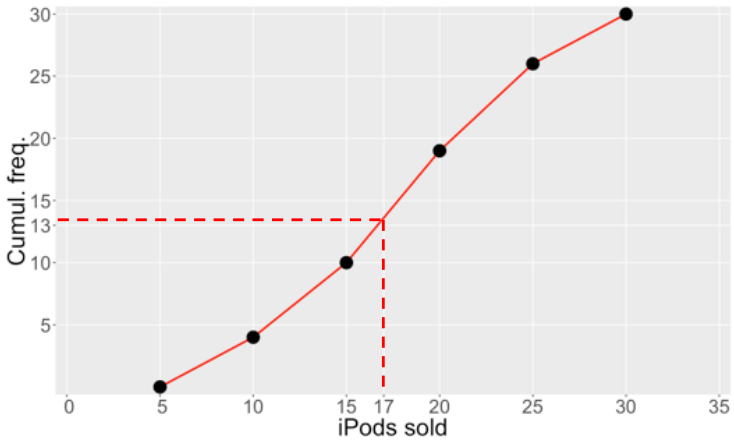


Figure 5.22: Line graph for the cumulative frequency distribution of the data set in Table 5.19. The dashed line is used to extrapolate a cumulative frequency not reported in the original frequency table.

the number of days for which 17 or fewer iPods were sold, which gives ≈ 13, as shown by the dashed line.

Exercise 5.2.7 *The data shown in Table 5.20 represent the frequency distribution of ages for all 50 employees of a company. Do all classes have the same width? If yes, what is that width? Calculate relative frequency and percentage distributions. Prepare a cumulative frequency distribution table (including cumulative relative frequencies and percentages). What percentage of the employees of this company are age 43 or younger? And what percentage of the employees are 44 or older? Draw a line graph for the cumulative percentage distribution and use it to find the percentage of employees who are age 40 or younger.*

Age	Number of employees
18 - 30	12
31 - 43	19
44 - 56	14
57 - 69	5

Table 5.20: Frequency distribution of ages in a company.

5.2.6 More tools for quantitative data (I). Stem-and-leaf displays.

Another technique that can be used to present quantitative data in condensed form is the **stem-and-leaf display**. An advantage of a stem-and-leaf display over a frequency distribution is that by preparing a stem-and-leaf display we do not lose information on individual observations. A stem-and-leaf display can be constructed only for quantitative data.

An example can easily clarify how to construct a stem-and-leaves display. Consider the scores of 30 students on a test, shown in Table 5.21.

Scores (0-100)									
75	52	80	96	65	79	71	87	93	95
69	72	81	61	76	86	79	68	50	92
83	84	77	64	71	87	72	92	57	98

Table 5.21: Raw quantitative data: test scores of 30 students.

Each score is split into two parts. The first part contains the first digit, which is called the **stem**. The second part contains the second digit, which is called the **leaf**. Thus, for the score of the first student, which is 75, 7 is the stem and 5 is the leaf. For the score of the second student, which is 52, the stem is 5 and the leaf is 2. One can observe from the data that the stems for all scores are 5, 6, 7, 8, and 9 because all the scores lie in the range 50 to 98. The stems are arranged in increasing order and marked on the left side of a vertical bar, as shown in Figure 5.23.

```
5 | 027
6 | 14589
7 | 112256799
8 | 0134677
9 | 223568
```

Figure 5.23: Stem and leaves graph for the data set of Table 5.21.

After listing the stems, the leaves for all scores are recorded next to the corresponding stems on the right side of the vertical line. For example, for the first score the leaf 5 must be written next to the stem 7; for the second score the leaf 2 must appear next to the stem 5. And so forth. The complete stem-and-leaf display should then appear as shown in Figure 5.23, where the leaves for each stem have also been ranked in increasing order.

Stem-and-leaves displays show in a very clear and intuitive way how data are distributed, without loosing information on individual observations (unlike

frequency distributions). Stem-and-leaves displays are generated in R by the function stem(), as briefly shown in the following example of R code.

```
> df
   scores
1      75
2      52
3      80
4      96
5      65
6      79
...
> stem(df$scores,scale=0.5,width=80)

  The decimal point is 1 digit(s) to the right of the |

  5 | 027
  6 | 14589
  7 | 112256799
  8 | 0134677
  9 | 223568
```

Exercise 5.2.8 *Table 5.22 gives the total yards gained rushing during the 2009 season by 14 running backs of 14 college football teams. Prepare a stem-and-leaf display.*

Total yards gained						
745	921	1133	1024	848	775	800
1009	1275	857	933	1145	967	995

Table 5.22: Raw data: total yards gained in rushing actions in the 2009 season by 14 running backs.

5.2.7 More tools for quantitative data (II). Dot plots.

One of the simplest methods for graphing and understanding quantitative data is to create a **dotplot**. Dotplots are very useful for detecting **outliers** (also called **extreme values**) in a data set. Outliers are the values that are extremely large or extremely small with respect to the rest of the data values.

Figure 5.24 lists the lengths of the longest field goals (in yards) made by all kickers in the american National Football League (NFL) during the 2008 season. Examining the dotplot of Figure 5.24 reveals that there are two clusters (groups) of data. Approximately 70% of the kickers made field goals of 50 to 57 yards. All but one of the remaining kickers completed long field goals of 43 to 47 yards. The isolated point at 26 yards is likely to be an outlier.

The following is the R code used to generate Figure 5.24.

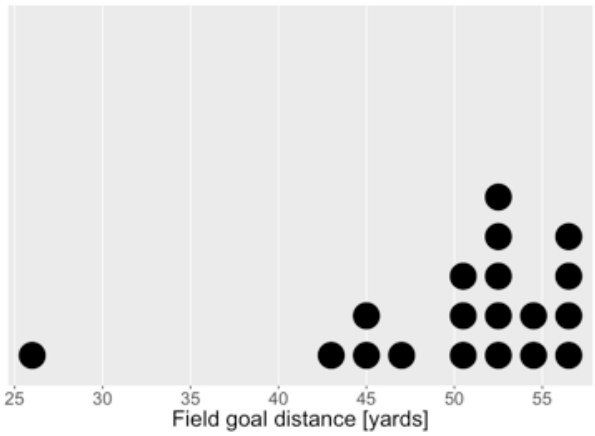


Figure 5.24: Dotplot of the longest field goal lengths for the NFL season 2008.

```
#### Create data frame with raw data
> df <- data.frame(kicks=c(54, 47, 53, 45, 26, 56, 56, 53,
    52, 53, 45, 43, 50, 50, 55, 57, 53, 57, 51))
#### Plot
> ggplot(df, aes(x = kicks)) + geom_dotplot(binwidth =
    1.5,stackratio=1.5,dotsize=1.0)
   +scale_y_continuous(NULL, breaks = NULL)
   + theme(text = element_text(size=20))
   + xlab("Field goal distance [yards]")+
      scale_x_discrete(limits=c(25,30,35,40,45,50,55,60))
```

Exercise 5.2.9 *The data in Table 5.23 give the number of times each of 30 randomly selected account holders at a bank used that banks ATM during a 60-day period. Create a dotplot for these data and point out any clusters or outliers.*

Number of ATM operations									
3	2	3	2	2	5	0	4	1	3
2	3	3	5	9	0	3	2	2	15
1	3	2	7	9	3	0	4	2	2

Table 5.23: Raw data: number of times a bank's ATM was used by account holders.

5.3 Numerical descriptive measures

In Section 5.2 we discussed how to summarize and display data using tables and graphs. Here we shall discuss how to describe **numerically** the main characteristics of a data set. **Numerical summary measures** are essential components of any statistical data analysis process. The measures discussed in this chapter include measures of i) **central tendency**, ii) **dispersion** (or **spread**), and iii) **position**.

5.3.1 Measures of central tendency

A **measure of central tendency** is a single value that attempts to describe a data set by identifying the central position within that data set. Measures of central tendency are sometimes called **measures of central location** or **summary statistics**. The **mean** (often called the average) is most likely the most popular measure of central tendency, but there are others, such as the **median** and the **mode**. The mean and the median apply only to quantitative data, whereas the mode can be used with either quantitative and qualitative data.

The mean, median and mode are obviously all valid measures of central tendency, but under different conditions some measures of central tendency become more appropriate than others. The mode should be used when calculating measure of center for qualitative variables. When the variable is quantitative with a symmetric distribution, then the mean is a proper measure of center. In case of a quantitative variable with skewed distribution, the median is a good choice for the measure of center. This is related to the fact that the mean can be highly influenced by **outliers**.

Mean

The mean (or average) is the most popular and well known measure of central tendency. It can be used with both discrete and continuous data.

The mean is equal to the sum of all the values in the (ungrouped) data set divided by the number of values in the data set. So, if we have n values in a data set and they have values $x_1, x_2, ..., x_n$, the sample mean is given by

$$\bar{x} = \frac{x_1 + x_2 + ... + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}\,. \tag{5.3}$$

The mean calculated for **sample data** is usually denoted by $\bar{x}$ as shown in Eq. 5.3, while the mean calculated for **population data** is denoted by the Greek letter μ and the population size is denoted by N.

The mean is essentially the most common value of a data set, although **it happens often that it is not one of the actual values observed in the data set**.

In R the mean can easily be implemented in one line, or can be calculated using the built-in function mean().

```
#### Two equivalent ways of calculating the mean in R
> x = c(1,2,3,4,5,3,4,2,100)
#### User-defined function
mymean <- function(arg) {
  l = length(arg)
  m = sum(arg)/l
  return(m)
}
> mymean(x)
[1] 13.77778
#### Built-in function
> mean(x)
[1] 13.77778
```

Exercise 5.3.1 *Table 5.24 lists the total sales (rounded to $ billions) of six US companies for the year 2008. Find the 2008 mean sales for these six companies.*

Company	**Sales** (billion $)
General Motors	149
Wal-Mart Stores	406
General Electric	183
Citigroup	107
Exxon Mobil	426
Verizon Communication	97

Table 5.24: Sales of six US companies for 2008.

Exercise 5.3.2 *Table 5.25 shows the ages (in years) of all eight employees of a small company. Find the mean age of these employees.*

Age of 8 employees							
53	32	61	27	39	44	49	57

Table 5.25: Age of the eight employees of a company.

Exercise 5.3.3 *Consider again the data set of Table 5.25. Select two distinct samples, each of 3 employees, and calculate their mean values $\bar{x}$. Verify that sample means for the same population do not necessarily coincide.*

The mean has one main disadvantage, that is, it is particularly susceptible to the influence of outliers. These are values that are unusual compared to the rest of the data set by being especially small or large. For example, consider the wages of the employees of a factory (Table 5.26).

Employee	1	2	3	4	5	6	7	8
Salary (USD)	15k	18k	16k	14k	15k	17k	90k	95k

Table 5.26: Wages of the employees of a factory.

Exercise 5.3.4 *Calculate the mean salary of the eight employees of Table 5.26. Is the result really representative of the typical salary in the company?*

The exercise 5.3.4 probably gave a mean salary of $35k (that is, if you did the calculation correctly!). However, a simple visual inspection of the raw data suggests that this mean value might not be the best way to accurately represent the typical salary of a worker, as most workers have salaries in the 14k to 18k range. The mean is clearly being skewed by the last two large salaries (probably the managers). Therefore, in this situation, we would like to have a better measure of central tendency. As we will find out later, taking the median would be a better measure of central tendency.

Let us consider another simple example to further clarify the effect of outliers on the mean. Table 5.27 lists the total philanthropic donations (in million dollars) by six American companies during the year 2007.

Corporation	**Money given** (million $)
CVS	22.4
Best Buy	31.8
Staples	19.8
Walgreen	9.0
Lowe's	27.5
Wal-Mart	337.9

Table 5.27: Philanthropic donations by 6 American companies in 2007.

Notice that the charitable contributions made by Wal-Mart are very large compared to those of other companies. Hence, it is an outlier.

Exercise 5.3.5 *Calculate the mean donation for the companies listed in Table 5.27* ***with and without*** *the outlier (Wal-Mart).*

Exercise 5.3.5 clearly shows that including the contribution of Wal-Mart causes more than a threefold increase in the value of the mean. This can obviously lead to completely misleading conclusions!

Median

The **median** is the **middle score** for a set of data that has been **arranged in increasing order of magnitude**. In other words, **it divides a ranked data set into two equal parts**.

Consider for example the data set of Table 5.28, representing the ages of the eleven employees of a small start-up company.

Age of 11 employees - Raw data										
25	22	24	21	18	21	23	21	26	64	63

Table 5.28: Age of the (eleven) employees of a small company.

We first need to rearrange that data into ascending order (that is, smallest first), as shown in Table 5.29.

Age of 11 employees - Sorted										
18	21	21	21	22	**23**	24	25	26	63	64

Table 5.29: Age of the (eleven) employees of a small company, sorted in increasing order. The median is highlighted in bold.

The median is in this case 23 (pretty young company!). It is obviously the middle value because there are 5 values before it and 5 values after it. This works fine with an odd number of data, but what happens with an even number, if the start-up had only 10 employees for instance? In that case one should simply take the middle two values (the 5^{th} and 6^{th}) and average the result. So, for instance, let us get rid of the 26 year old employee of Table 5.28. The sorted data set turns out to be as shown in Table 5.30. The average of the 5^{th} and 6^{th} values gives $(22 + 23)/2 = 22.5$, which is the median of the new data set.

Age of 10 employees - Sorted									
18	21	21	21	**22**	**23**	24	25	63	64

Table 5.30: Age of the (ten) employees of a small company, sorted in increasing order. The median is given by the average of the two values highlighted in bold.

In R the median can be implemented as follows,

```
mymedian = function(v){
  n=length(v)
  w=sort(v,decreasing = FALSE)    # Rank vector
  if((n %% 2) == 1) {
    out = w[ceiling(n/2)]    # Middle value, odd length
  } else {
    out = mean(c(w[n/2],w[n/2+1])) # Middle value, even
        length
  }
  return(out)
}
```

or it can be calculated using the built-in function median(),

```
> x = c(1,2,3,4,5,3,4,2,100)
> mean(x)
[1] 13.77778
> median(x)
[1] 3
```

The median gives the center of a histogram, with half of the data values to the left of the median and half to the right. **The median is much less affected by outliers and skewed data than the mean**. As you may have noticed in Table 5.30, two employees, probably the founders of the start-up, are much older (63 and 64) than the others. As a consequence, using the mean as a measure of central tendency would suggest an average age of 30.2 (*you can verify it!*). Not quite as young as the median 22.5 would suggest! **The median should be preferred over the mean as a measure of central tendency for data sets that contain outliers or for skewed data sets**.

Let us consider one more example. Table 5.31 shows the profits for the year 2008 of 12 large companies in billions of dollars.

Company	**2008 profit** (billion $)
Merck & Co.	8
IBM	12
Unilever	7
Microsoft	17
Petrobras	14
Exxon Mobil	45
Lukoil	10
AT&T	13
Nestl	17
Vodafone	13
Deutsche Bank	9
China Mobile	11

Table 5.31: Profits for 12 companies for 2008.

Note that one of the companies, Exxon Mobil, made an especially large profit ($45 billion) compared to all other companies whose profits were in the range $\$7 - 17$ billion. Because of this "outlier", the median might be the best choice as a measure of central tendency.

Note. The "skewness" of a distribution can be quantified in R using the function skewness(), which requires the library "e1017". It will suffice here to say that the skewness can be calculated using various algorithms that may lead to slightly varying results. The R function skewness() implements three different methods, specified by the argument *type*. The skewness can be positive

or negative, depending on whether the distribution is skewed to the left or to the right. It is zero for symmetric distributions. The data of Table 5.31 for instance yield a skewness of ≈ 2.87, as shown in the R code below.

```
# IMPORTANT: Skewness requires library "e1071"
> library(e1071)

# Symmetric distribution
> x <- rnorm(100000)
> skewness(x)
[1] 0.003843261

# Skewed distribution
> df =
    data.frame(profit=c(8,12,7,17,14,45,10,13,17,13,9,11))
> skewness(df$profit,type=2)
[1] 2.869023
```

Exercise 5.3.6 *Calculate mean and median for the profits listed in Table 5.31.*

Exercise 5.3.7 *Table 5.32 gives the prices (in some currency) of seven houses sold last month in a city. Find mean and median.*

Price ($\times$1000)						
312	257	421	289	526	374	497

Table 5.32: Prices of sold houses.

Exercise 5.3.8 *The data in Table 5.33 give the 2006-07 team salaries for 20 teams of the English Soccer Premier League. The salaries are given in the order in which the teams finished the 2006-07 season and are in millions of British pounds. Find the mean and median for these data.*

Team salaries (£millions)									
92.3	132.8	77.6	89.7	43.8	38.4	30.7	29.8	36.9	36.7
43.2	38.3	62.5	36.4	44.2	35.2	27.5	22.4	34.3	17.6

Table 5.33: Team salaries in the English Soccer premier League. Source: BBC, May 28, 2008.

Exercise 5.3.9 *The data in Table 5.34 give the 2008 profits (in $ millions) of six Arizona-based companies for the year 2008. The data represent the following companies, respectively: Freeport-McMoRan Copper & Gold, Avnet, US Airways Group, Allied Waste Industries, Insight Enterprises, and PetSmart. Find the mean and median for these data.*

Profits ($ millions)					
2977.0	393.1	427.0	273.6	77.8	258.7

Table 5.34: Profits of six Arizona-based companies. Source: Fortune, May 5, 2008.

Exercise 5.3.10 *The data in Table 5.35 represent the numbers of tornadoes that touched down during 1950 to 1994 in the 12 American states that had the most tornadoes during this period. The data for these states are given in the following order: CO, FL, IA, IL, KS, LA, MO, MS, NE, OK, SD, TX.*

- *Calculate the mean and median for these data.*
- *Identify the outlier in this data set. Drop the outlier and recalculate the mean and median. Which of these two summary measures changes by a larger amount?*
- *Which is the better summary measure for these data, the mean or the median? Explain.*

Number of tornadoes (1950-1994)					
1113	2009	1374	1137	2110	1086
1166	1039	1673	2300	1139	5490

Table 5.35: Number of tornadoes during the period 1950-1994 for 12 American states. Source: Storm Prediction Center, 2009.

Mode

The mode is the value that occurs with the highest frequency in a data set. Consider for instance the speeds in miles/hour of eight cars that were stopped for speeding violations, as shown in Table 5.36.

Speeds (miles/hour)							
77	82	74	81	79	84	74	78

Table 5.36: Speeds of 8 cars stopped for speeding violations.

In this data set, 74 occurs twice, and each of the remaining values occurs only once. Because 74 occurs with the highest frequency, it is the mode.

A major shortcoming of the mode is that a data set may have none or may have more than one mode, whereas it will always have only one mean and only one median. For instance, a data set with each value occurring only once has

no mode. A data set with only one value occurring with the highest frequency has only one mode. The data set in this case is called **unimodal**. A data set with two values that occur with the same (highest) frequency has two modes. The distribution, in this case, is said to be **bimodal**. If more than two values in a data set occur with the same (highest) frequency, then the data set contains more than two modes and it is said to be **multimodal**.

Unfortunately R does not provide a built-in function for calculating the mode. However, it is quite straightforward to produce a user-defined function that would accomplish the task.

```
getmode = function(v) {
  uniqv = unique(v)   # v without double elements
  occur = tabulate(match(v, uniqv)) # number of occurrences
                                    # of elements of v
  out = uniqv[which(occur==max(occur))]
  return(out)
}
```

Exercise 5.3.11 *Consider again the profits of Table 5.31. What is the mode? (Hint: the data set is **bimodal**)*

Exercise 5.3.12 *The ages of 10 randomly selected students from a class are shown in Table 5.37. Find the mode (or modes).*

Age									
21	19	27	22	29	19	25	21	22	30

Table 5.37: Age of 10 random students in a class.

Exercise 5.3.13 *Last years incomes of five randomly selected families are given in Table 5.38. Find the mode.*

Incomes (USD)				
76150	95750	124985	87490	53740

Table 5.38: Income of 5 families.

One advantage of the mode is that it can be calculated for both kinds of data, quantitative and qualitative, whereas the mean and median can be calculated for only quantitative data. The R function getmode() defined above works for both quantitative and qualitative data.

Exercise 5.3.14 *Table 5.39 shows the status of five students who are members of the student senate at an American college. What is the mode? Note that obviously we cannot calculate the mean and median for this data set.*

Students status				
senior	sophomore	senior	junior	senior

Table 5.39: Status of 5 students at an American college.

Exercise 5.3.15 *Due to antiquated equipment and frequent windstorms, a small town often suffers power outages. The data of Table 5.40 give the numbers of power outages for each of the past 12 months. Compute the mean, median, and mode for these data.*

Power outages / month											
4	5	7	3	2	0	2	3	2	1	2	4

Table 5.40: Power outages over the past 12 months.

Relationships among mean, median and mode

We briefly analyze here the relationships among the mean, median, and mode for different types of histograms or frequency distribution curves.

- For **symmetric** distributions with one peak, the values of **the mean, median, and mode are identical**, and they lie at the center of the distribution. This is shown in Figure 5.25.
- For distributions that are **skewed** to the right, as shown in Figure 5.26, the value of the mean is the largest, the mode is the smallest, and the value of the median lies between these two. The mode always occurs at the peak point. The value of the mean is the largest in this case because it is sensitive to outliers that occur in the tail. These outliers pull the mean to the right.
- Symmetric distributions with two peaks like that of Figure 5.27 are bimodal and exhibit coinciding mean and median values.

More about the mean

One property of the mean is that if we know the means and sample sizes of two (or more) data sets, we can calculate the **combined mean** of both (or all) data sets. The combined mean $\bar{x}$ for two data sets is calculated by using the formula,

$$\bar{x} = \frac{n_1\bar{x}_1 + n_2\bar{x}_2}{n_1 + n_2}\,, \tag{5.4}$$

where n_1 and n_2 are the sample sizes of the two data sets and $\bar{x}_1$ and $\bar{x}_2$ are the means of the two data sets, respectively.

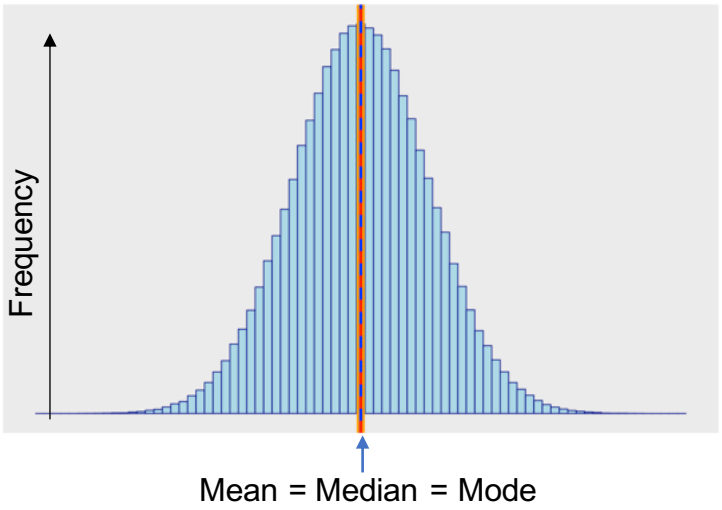


Figure 5.25: Mean (blue), median (red), and mode (orange) are identical for symmetric distributions with a single peak.

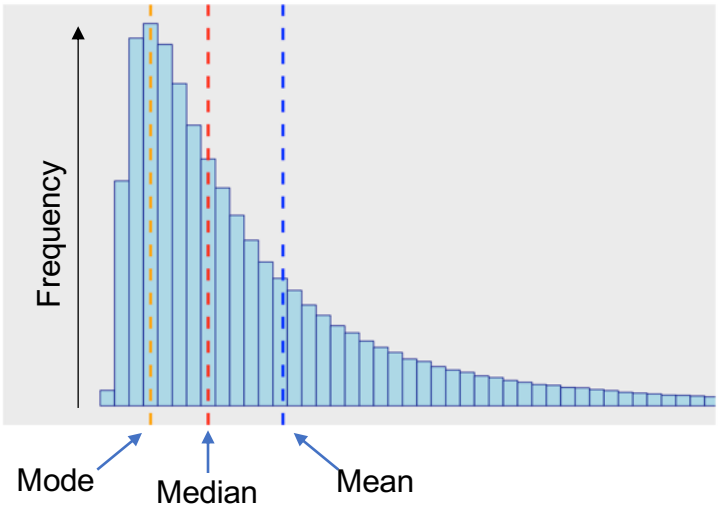


Figure 5.26: Mean (blue), median (red), and mode (orange) do not coincide for skewed frequency distributions.

Exercise 5.3.16 *Suppose that a sample of 10 statistics books yields a mean price of 140 CHF and a sample of 8 mathematics books yields a mean price of 160 CHF. What is the mean expense for a student who needs both statistics and mathematics books? (Hint: calculate a combined mean, with* $n_1 = 10$, $n_2 = 8$,

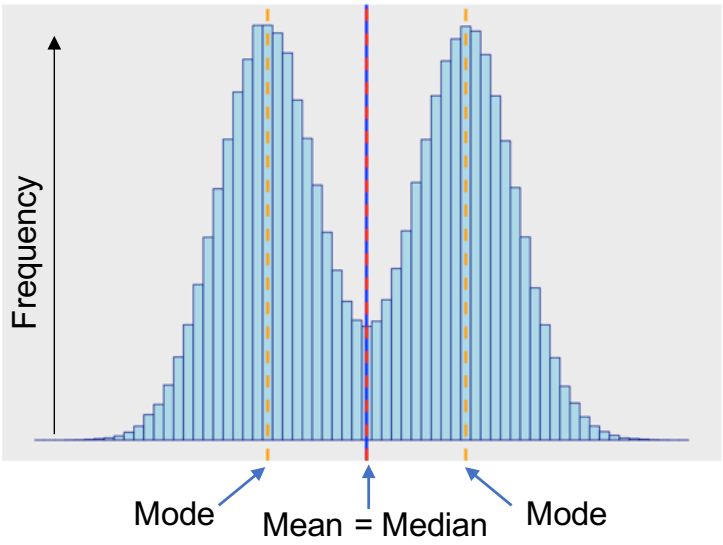


Figure 5.27: A bimodal frequency distribution with coinciding mean (blue) and median (red).

$\bar{x}_1 = 140$, $\bar{x}_2 = 160$*)*

The **trimmed mean** is calculated by dropping a certain percentage of values from each end of a ranked data set. The trimmed mean is especially useful as a measure of central tendency when a data set contains a few outliers at each end. For instance, to calculate the 10% trimmed mean of a data set, first of all the data values must be ranked in increasing order. Then 10% of the total number of values (that is, of the size of the data set) are dropped at both ends of the ranked data set. The mean of the remaining 80% of the values will give the sought 10% trimmed mean.

Note that the R function mean() provides by default a trim option.

Exercise 5.3.17 *Suppose that the data of Table 5.41 give the ages (in years) of 10 employees of a company. Calculate the mean, median and 10% trimmed mean. What do you notice? (Hint: Note that this data set contains 10 values, and 10% of 10 is 1. Thus, if one drops the smallest value and the largest value, the mean of the remaining 8 values will be called the 10% trimmed mean.).*

Age									
47	53	38	26	39	49	19	67	31	23

Table 5.41: Age of 10 employees of a company.

Note. If we had for instance a data set of only 9 elements, the 10% trimmed mean would coincide with the normal mean. Indeed, 10% of 9 is 0.9, and no elements are dropped from the original data set. In order to drop a single element at both sides of the ranked data set we would need a $1/9 \approx 0.12\%$ trimmed mean. In order to drop two elements, we would need a $2/9 \approx 0.23\%$ trimmed mean. And so forth. In general, in order to drop m elements from a data set containing n elements, we need a $(m/n * 100)\%$ trimmed mean.

```
> x = c(1,2,3,4,5,3,4,2,100)
> length(x)
[1] 9
> mean(x)
[1] 13.77778
> mean(x,trim=0.1)
[1] 13.77778
> mean(x,trim=0.12)
[1] 3.285714
> mean(x,trim=0.2)
[1] 3.285714
> mean(x,trim=0.23)
[1] 3.2
```

Exercise 5.3.18 *The data in Table 5.42 give the prices (in thousands of some currency) of 20 houses sold recently in a city. Find the 20% trimmed mean for this data set.*

House prices ($\times 1000$)									
184	297	365	309	245	387	369	438	195	390
323	578	410	679	307	271	457	795	259	590

Table 5.42: Prices for 20 houses in a city.

In some applications, certain values in a data set may be considered more important than others. For example, to determine students grades in a course, an instructor may assign a weight to the final exam that is twice as much as that to each of the other exams.

In such cases, it is more appropriate to use the **weighted mean**. In general, for a sequence of n data values $x_1, x_2, ..., x_n$ that are assigned weights $w_1, w_2, ..., w_n$, respectively, the weighted mean is given by,

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}. \tag{5.5}$$

In R the function weighted.mean() does the job.

```
> x = c(1,2,3,4,5,3,4,2,100)  # data set
> w = c(1,1,3,1,1,2,1,1,0.1) # weights
# Note: the outlier is assigned a very low weight!
> mean(x)
[1] 13.77778
> median(x)
[1] 3
> weighted.mean(x,w)
[1] 3.873874
```

Exercise 5.3.19 *Suppose that an instructor gives two intermediate exams and a final exam, assigning the final exam a weight twice that of each of the other exams. Find the weighted mean for a student who scores 73 and 67 on the first two exams and 85 on the final exam.*

When studying phenomena such as inflation or population changes that involve periodic increases or decreases, the **geometric mean** is used to find the average change over the entire period under study. To calculate the geometric mean of a sequence of n values $x_1, x_2, ..., x_n$, we multiply them together and then find the n^{th} root of this product. Thus,

$$\bar{x} = \sqrt[n]{x_1 \cdot x_2 \cdot ... \cdot x_n}\,. \tag{5.6}$$

Exercise 5.3.20 *Suppose that the inflation rates for the last five years are 4%, 3%, 5%, 6%, and 8%, respectively. Therefore, at the end of the first year the price index will be 1.04 times the price index at the beginning of the year, and so on. Find the mean rate of inflation over the 5-year period. (Hint: find the geometric mean of the data set 1.04, 1.03, 1.05, 1.06, and 1.08).*

5.3.2 Measures of dispersion

The measures of central tendency, such as the mean, median, and mode, do not reveal the whole picture of the distribution of a data set. **Two data sets with the same mean may have completely different spreads**. The variation among the values of observations for one data set may be much larger or smaller than for the other data set. Henceforth we will not make any distinction between the words dispersion, spread, and variation, and use them indifferently.

Consider for instance two data sets representing the age of the employees working for two small companies, as shown in Table 5.43. The mean age of

Age of the employees							
Company 1:	47	38	35	40	36	45	39
Company 2:		70	33	18	52	27	

Table 5.43: Age of the employees working for two companies.

workers in both these companies is the same, 40 years. If one did not know the ages of individual workers and was told only that the mean age of the workers at both companies is the same, one might be tempted to deduce that the workers at these two companies have a similar age distribution. As we can observe in Figure 5.28, however, the variation in the employees' ages for each of these two companies is very different. As illustrated in the diagram, the ages of the employees at the second company have a much larger variation than the ages of the employees at the first company.

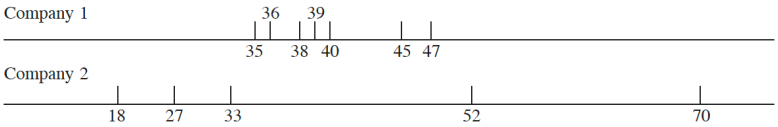


Figure 5.28: Dispersion of the employees' ages at a glance.

Thus, the mean, median, or mode are usually not a sufficient measure to reveal the shape of the distribution of a data set. We also need a measure that can provide some information about the variation among data values. Measures describing the spread of a data set are called **measures of dispersion**. The measures of central tendency and dispersion taken together give a better picture of a data set than the measures of central tendency alone.

The range

The **range** is the simplest measure of dispersion to calculate. It is obtained by taking **the difference between the largest and the smallest values** in a data set.

```
> x = c(1,2,3,4,5,3,4,2,100)
> range(x)
[1]   1 100
> range(x)[2]-range(x)[1]
[1] 99
```

Exercise 5.3.21 *Table 5.44 gives the total areas in square miles of four states of the USA. Find the range.*

The range simply defines the interval within which the observed data fall. And it is not a very satisfactory measure of dispersion. Indeed, the range, just like the mean, has the disadvantage of being influenced by outliers. In the example of Table 5.44, if the state of Texas is dropped, the range decreases from 217626 square miles to only 20252 square miles. A factor ten!

Consequently, the range is definitely not a good measure of dispersion when a data set contains outliers. Another disadvantage of using the range as a measure of dispersion is that its calculation is based on two values only, that is, the largest and the smallest. All other values in a data set are ignored, and a lot of information is thus lost.

State	Area (square miles)
Arkansas	53182
Louisiana	49651
Oklahoma	69903
Texas	267277

Table 5.44: Total area of four American states.

Variance and Standard Deviation

The **standard deviation** is the most used measure of dispersion. The value of the standard deviation is a measure of how closely the values of a data set are clustered around the mean.

The standard deviation is defined as the **square root of the variance**. It is standard practice in statistics to distinguish between the variance calculated for **population data**, denoted by $\boldsymbol{\sigma^2}$, and the variance calculated for sample data, denoted by $\boldsymbol{s^2}$. The variance is defined as,

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N} \quad \text{and} \quad s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}, \tag{5.7}$$

where μ and N are the population mean and size, respectively, and $\bar{x}$ and n are the sample mean and size, respectively. Sample variance and standard deviation can be implemented in R as follows (it is straightforward to implement analogous functions for populations).

```
# Sample variance
var = function(v) {
  n = length(v)
  out = sum((v-mean(v))^2) / (n-1)
  return(out)
}

# Sample standard deviation
sd = function(v) {
  n = length(v)
  out = sqrt(sum((v-mean(v))^2) / (n-1))
  return(out)
}
```

The quantities $x_i - \mu$ and $x_i - \bar{x}$ in Eq. 5.7 represent the **deviation** of the i^{th} value x_i of the data set x from the mean.

One might wonder why in Eq. 5.7 the denominator for the population variance is N, while for the sample variance it is $n-1$. The reason is that the sample variance underestimates the population variance when using a denominator equal to n. We will not go into more details here.

Exercise 5.3.22 *Suppose that the midterm scores of a sample of four students are 82/100, 95/100, 67/100, and 92/100. Find the mean and standard deviation.*

From a computational point of view it might be easier to use shortcut formulas to estimate variance and standard deviation. It can be shown that the following formulas are equivalent to Equations 5.7,

$$\sigma^2 = \frac{\sum_{i=1}^{N} x_i^2 - \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N}}{N} \quad \text{and} \quad s^2 = \frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}\,. \tag{5.8}$$

Exercise 5.3.23 *Implement Eqs. 5.8 in R.*

Exercise 5.3.24 *Table 5.45 gives the 2008 market values (rounded to billions of dollars) of five international companies. Find mean and median, then calculate variance and standard deviation using Eqs. 5.8.*

Company	**Market values**
	(billions of dollars)
PepsiCo	75
Google	107
PetroChina	271
Johnson & Johnson	138
Intel	71

Table 5.45: Market values of 5 companies for 2008.

Exercise 5.3.25 *Table 5.46 shows the net 2009 earnings (in thousands of CHF) for all six employees of a small company. Calculate mean, median, and standard deviation for these data.*

Earnings (×1000 CHF)					
88.50	108.40	65.50	52.50	79.80	54.60

Table 5.46: Earnings for all six employees of a company.

Exercise 5.3.26 *Can the standard deviation have a negative value?*

Exercise 5.3.27 *The following data set belongs to a population,*

$$5, \quad -7, \quad 2, \quad 0, \quad -9, \quad 1, \quad 61, \quad 7\,.$$

Calculate mean, median, range, variance, and standard deviation.

Exercise 5.3.28 *The following data give the number of shoplifters apprehended during each of the past 8 weeks at a large department store.*

$$7, \quad 1, \quad 8, \quad 3, \quad 1, \quad 51, \quad 26, \quad 1, \quad 1\,.$$

- *Find the mean for these data.*
- *Calculate the deviations of the data values from the mean. Is the sum of these deviations zero?*
- *Find the median. Is it different from the mean? Why?*
- *Calculate the range, variance, and standard deviation.*

Exercise 5.3.29 *Consider again Exercise 5.3.10 about the number of tornadoes that touched down in 12 American states during the period 1950-1994 (see Table 5.35). We have already calculated mean and median. Now find range, variance and standard deviation (including outliers).*

Exercise 5.3.30 *Consider the two data sets of Table 5.47. Note that each value of the second data set is obtained by adding 7 to the corresponding value of the first data set. Calculate the mean and standard deviation for each of these two data sets using the formula for sample data. Comment on the relationship between the two means and standard deviations.*

Data set 1	12	25	37	8	41
Data set 2	19	32	44	15	48

Table 5.47: Two data sets, in some units.

Exercise 5.3.31 *Consider now the two data sets of Table 5.48. Note that each value of the second data set is obtained by multiplying the corresponding value of the first data set by 2. Calculate the mean and standard deviation for each of these two data sets using the formula for population data. Comment on the relationship between the two means and standard deviations.*

Data set 1	4	8	15	9	11
Data set 2	8	16	30	18	22

Table 5.48: Other two data sets, in some units.

Note. The R functions var() and sd() refer to the sample variance and standard deviation, respectively. There are no corresponding functions for populations. A possible quick solution would be to define our own population variance as follows (the standard deviation is then its square root).

```
popvar = function(v) {
  n = length(v)
  out = var(v)*(n-1)/n
  return(out)
}
```

5.3.3 Descriptive measures for grouped data

We learned in Section 5.3.1 that the **mean** is obtained by dividing the sum of all values by the number of values in a data set. However, if the data are given in the form of a frequency table, we no longer know the values of individual observations. In such cases, one can find an approximation to the sum of the individual values using the following equation,

$$\bar{x} = \frac{\sum_i m_i \cdot f_i}{n}\,, \tag{5.9}$$

where m and f are a class midpoint (defined in Section 5.2.3) and frequency, respectively, the sum is calculated over all classes and n is the size of the data set.

Exercise 5.3.32 *The frequency table 5.49 shows the daily commuting times of all 25 employees of a company, with the midpoints for each class. Find the mean using Eq. 5.9. (Hint: note that since we are talking about all the employees of a company, we are talking about population data. In this case in Eq. 5.9 one should replace $\bar{x}$ with μ and n with N.)*

Daily commuting times (minutes)	**Number of employees** (f)	**Midpoint** (m)
0 to less than 10	4	5
10 to less than 20	9	15
20 to less than 30	6	25
30 to less than 40	4	35
40 to less than 50	2	45

Table 5.49: Daily commuting times for the 25 employees for a company.

It turns out that the employees of Table 5.49 spend an average of 21.40 minutes a day commuting from home to work. However, there are employees who commute less than 10 minutes per day, and others that spend everyday more than 40 minutes commuting to work. We know that this feature of the data set can be captured by quantifying the spread of the data. So, one can define the **variance** as,

$$\sigma^2 = \frac{\sum_i f_i\,(m_i - \mu)^2}{N} \quad \text{and} \quad s^2 = \frac{\sum_i f_i\,(m_i - \bar{x})^2}{n-1}\,, \tag{5.10}$$

for grouped population and sample data, respectively. The **standard deviation** is calculated as usual as the square root of the variance, that is, $\sigma = \sqrt{\sigma^2}$ and $s = \sqrt{s^2}$.

Exercise 5.3.33 *Calculate the standard deviation for the data of Table 5.49 using Eq. 5.10.*

For practical purposes, one can use the following equations (analogous to Eqs. 5.10) to calculate population and sample variance, respectively,

$$\sigma^2 = \frac{\sum_i m_i^2 f_i - \frac{\left(\sum_i m_i f_i\right)^2}{N}}{N} \quad \text{and} \quad s^2 = \frac{\sum_i m_i^2 f_i - \frac{\left(\sum_i m_i f_i\right)^2}{n}}{n-1}. \qquad (5.11)$$

Needless to say, the standard deviations are calculated by taking the square root of Eqs. 5.11.

Exercise 5.3.34 *Consider again the data of Table 5.49. Find variance and standard deviation using Eq 5.11.*

Exercise 5.3.35 *For 50 airplanes that arrived late at an airport during a week, the time by which they were late was observed. This is shown in Table 5.50. Find the mean, variance, and standard deviation.*

Delay (minutes)	**Number of airplanes** (f)
0 to less than 20	14
20 to less than 40	18
40 to less than 60	9
60 to less than 80	5
80 to less than 100	4

Table 5.50: Observed delays for 50 airplanes.

5.3.4 Meaning of the standard deviation

The mean and standard deviation can provide important information on the **proportion or percentage of the total observations that fall within a given interval about the mean**.

First, **Chebyshevs theorem** states that **for any number k greater than 1, at least $1 - 1/k^2$ of the data values lie within k standard deviations of the mean**.

This is illustrated in Figure 5.29.

Thus, for example, consider $k = 2$. Then, $1 - 1/k^2 = 1 - 1/4 = 0.75$ or equivalently 75%. Therefore, according to Chebyshevs theorem, at least 75% of the values of a data set lie within two standard deviations of the mean.

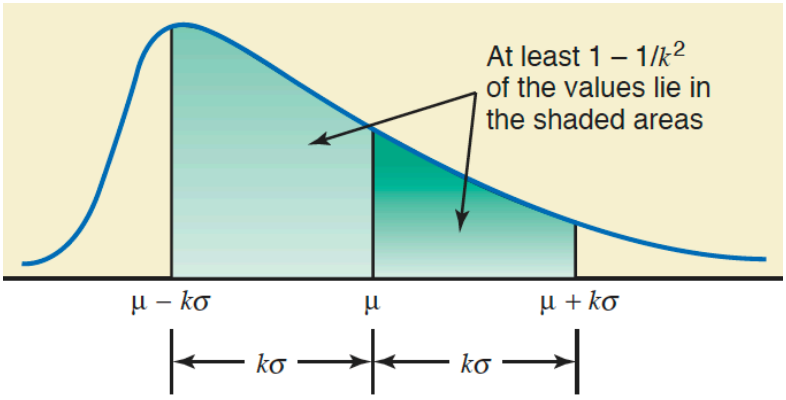


Figure 5.29: Chebyshev theorem.

Moreover, if $k = 3$, one has $1 - 1/k^2 = 89\%$. In other words, at least 89% of the data values fall within three standard deviations of the mean. And so forth.

Chebyshev's theorem can be applied to any kind of distribution.

For example, consider the following case study. The average systolic blood pressure for 4000 women who were screened for high blood pressure was found to be 187 mmHg with a standard deviation of 22 mmHg. Let us use Chebyshevs theorem to find at least what percentage of women in this group have a systolic blood pressure between 143 and 231 mmHg. From the given data one has $\mu = 187$ and $\sigma = 22$ mmHg. Each of the two points of interest, 143 and 231, is 44 units away from the mean μ. Therefore, one finds $k = 44/\sigma = 2$. Hence, according to Chebyshevs theorem, at least 75% of the women have systolic blood pressure between 143 and 231 mmHg.

```
> mu = 187; sigma = 22 # mmHg
> (231- mu) == (mu - 143)  # check for symmetric interval
[1] TRUE
> d = 231 - mu; d          # distance from mean
[1] 44
> k = d/sigma; k           # number of st. deviations
[1] 2
> cheb = 1-(1/k)^2; cheb   # Chebyshev's theorem
[1] 0.75
```

Alternatively, an **empirical rule** can be applied to bell-shaped distributions, as shown in Figure 5.30.

The rule states that,

- 68% of the observations lie within **1 standard deviation** of the mean.
- 95% of the observations lie within **2 standard deviations** of the mean.
- 99.7% of the observations lie within **3 standard deviations** of the mean.

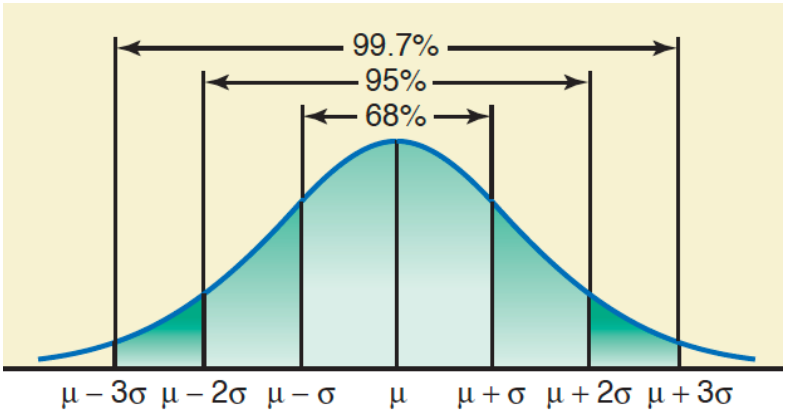


Figure 5.30: Empirical rule.

As an example, consider the following situation. The age distribution of a sample of 5000 persons is bell shaped with a mean of 40 years and a standard deviation of 12 years. Let us find the approximate percentage of people who are 16 to 64 years old. From the data, one has $\bar{x} = 40$ and $s = 12$ years. Each of the two points, 16 and 64, is 24 units away from the mean. Dividing 24 by 12, one can convert the distance between each of the two points and the mean in terms of standard deviations, that is, $2s$ in this case. The area of interest is thus the area between $\bar{x} - 2s$ and $\bar{x} + 2s$. Hence, approximately 95% of the people in the sample are 16 to 64 years old.

Exercise 5.3.36 *The mean time taken by all participants to run a road race was found to be 220 minutes with a standard deviation of 20 minutes. Using Chebyshevs theorem, estimate the percentage of runners who ran this road race in, 1) 180 to 260 minutes, 2) 160 to 280 minutes, 3) 170 to 270 minutes.*

Exercise 5.3.37 *Suppose that the prices of university textbooks follow a bell-shaped distribution with a mean of 105 CHF and a standard deviation of 20 CHF. Using the empirical rule, find the percentage of all textbooks with their prices between 1) 85 and 125 CHF, 2) 65 and 145 CHF. Find also the interval that contains the prices of 99.7% of college textbooks.*

5.3.5 Measures of position

A **measure of position** determines the position of a single value in relation to other values in a data set. Although there are many measures of position, only **quartiles**, **percentiles** and **percentile rank** are discussed in this section.

Quartiles and interquartile range

Quartiles are the summary measures that divide a ranked data set into **four equal parts**. These measures are the **first quartile** (denoted by Q_1), the

second quartile (denoted by Q_2), and the **third quartile** (denoted by Q_3). Note that the data should be ranked in increasing order before the quartiles are determined. A formal definition of quartiles can be given as follows.

Quartiles are three summary measures that divide a ranked data set into four equal parts. The second quartile is the same as the median of a data set. The first quartile is the value of the middle term among the observations that are less than the median, and the third quartile is the value of the middle term among the observations that are greater than the median.

The concept of quartiles is illustrated in Figure 5.31.

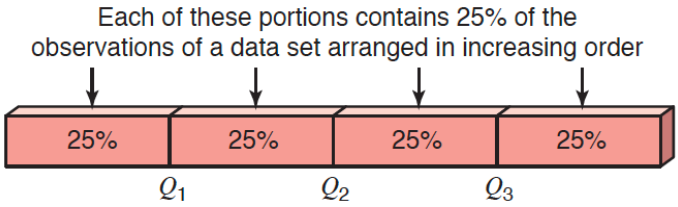


Figure 5.31: Quartiles.

Approximately 25% of the values in a ranked data set are less than Q_1 and about 75% are greater than Q_1. The second quartile, Q_2, divides a ranked data set into two equal parts; hence, the second quartile and the median are the same. Approximately 75% of the data values are less than Q_3 and about 25% are greater than Q_3. The difference between the third quartile and the first quartile for a data set is called the **interquartile range** (***IQR***). Thus, $IQR = Q_3 - Q_1$.

Let us consider again the example of Table 5.31 and find out both the values of the three quartiles and where the profit of Merck & Co falls in relation to these quartiles. The Table 5.31 is reproduced below as Table 5.51.

First, data should be ranked in increasing order, then the quartiles can be evaluated as sketched in Figure 5.32.

The value of Q_2, which is also the median, is given by the value of the middle term in the ranked data set, that is, Q_2 is \$12.5 billion.

The value of Q_1 is given by the value of the middle term (i.e., the median) of the values that fall below Q_2. In this case, Q_1 is \$9.5 billion.

Finally, the value of Q_3 is given by the value of the middle term (i.e., the median) of the values that fall above Q_2. For the data of this example, Q_3 is thus \$15.5 billion.

The value Q_1 = \$9.5 billion indicates that 25% of the companies in this sample had 2008 profits less than \$9.5 billion and 75% of the companies had 2008 profits higher than \$9.5 billion. Similarly, we can state that half of these companies had 2008 profits less (or greater) than \$12.5 billion. The value Q_3 = \$15.5 billion indicates instead that 75% of the companies had 2008 profits less than \$15.5 billion and of course 25% had profits greater than this value.

Company	**2008 profit** (billion $)
Merck & Co.	8
IBM	12
Unilever	7
Microsoft	17
Petrobras	14
Exxon Mobil	45
Lukoil	10
AT&T	13
Nestl	17
Vodafone	13
Deutsche Bank	9
China Mobile	11

Table 5.51: Profits for 12 companies for 2008.

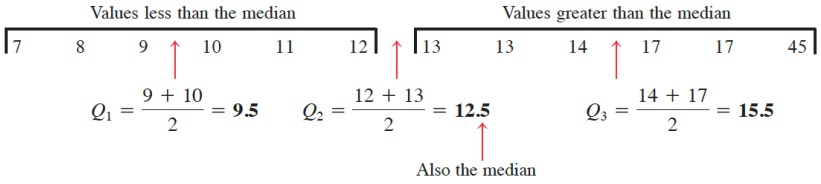


Figure 5.32: Quartiles for the data set of Table 5.51.

By looking at the position of $8 billion, which is the 2008 profit of Merck & Co, we can state that this value lies in the **bottom 25%** of the profits for 2008.

Percentile and percentile rank

Percentiles are the summary measures that divide a ranked data set into 100 equal parts. Each (ranked) data set has 99 percentiles that divide it into 100 equal parts. The k^{th} percentile is denoted by P_k, where k is an integer in the range 1 to 99. For instance, the 25th percentile is denoted by P_{25}. Figure 5.33 shows the positions of the 99 percentiles.

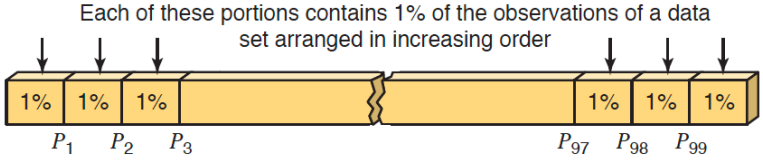


Figure 5.33: Percentiles.

Thus, the k^{th} percentile, P_k, can be defined as a value in a data set such that about $k\%$ of the measurements are smaller than the value of P_k and about $(100-k)\%$ of the measurements are greater than the value of P_k.

The approximate value of the k^{th} percentile is determined as,

$$P_k = \text{value of the } \left(\frac{k\cdot n}{100}\right)^{\text{th}} \text{ term in a ranked data set}\,, \tag{5.12}$$

where k denotes the number of the percentile and n is the sample size.

Let us find for example the 42nd percentile of the data of Table 5.51. The ranked data look like,

$$7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 13 \quad 14 \quad 17 \quad 17 \quad 45\,.$$

The position of the 42nd percentile can then be estimated using Eq. 5.12, that is,

$$\frac{k\cdot n}{100} = \frac{42\cdot 12}{100} = 5.04 \approx 5\,. \tag{5.13}$$

Note that the number given by Eq. 5.12 is rounded to the nearest integer. Therefore, the 42nd percentile is given by the 5th value in the ranked dataset, which yields $P_{42} = \$11$ billion. The information provided by this number essentially tells us that approximately 42% of the 12 companies of Table 5.51 had 2008 profits less than or equal to \$11 billion.

The **percentile rank** of a particular value x_i of a data set $\boldsymbol{x}$ gives the percentage of values in the data set that are less than x_i. The percentile rank can be calculated using the (rather intuitive) formula,

$$\text{Percentile rank of } x_i = \frac{\text{number of values less than } x_i}{n}\cdot 100\,, \tag{5.14}$$

where n is the total number of values in the data set.

So, for example, referring once more to Table 5.51, one can find the percentile rank for the \$14 billion profit of Petrobras as follows. Let us first consider again the ranked data set,

$$7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 13 \quad 14 \quad 17 \quad 17 \quad 45\,.$$

One can observe that 8 of the $n = 12$ values are less than \$14 billion. Hence, using Eq. 5.14,

$$\text{Percentile rank of } 14 = \frac{8}{12}\cdot 100 = 66.67\%\,.$$

Rounding this answer to the nearest integer value, we can state that about 67% of the 12 companies had less than \$14 billion profits in 2008. Or, 33% of the 12 companies had \$14 billion or higher profits in 2008.

Important note. In descriptive statistics, **quantiles** Q_k are often used instead of percentiles P_k. Here it will suffice to say that they are identical,

except for the fact that percentiles are expressed on a scale $0 - 100$ (that is, as percents) whereas quantiles are expressed on a scale $0 - 1$. So, $P_k = Q_{k/100}$.

Quartiles and percentiles (or quantiles) for the data of Table 5.51 can be calculated in R as follows.

```
> x = c(8,12,7,17,14,45,10,13,17,13,9,11) # raw data set
> y = sort(x); y
 [1]  7  8  9 10 11 12 13 13 14 17 17 45  # sorted
> median(y)                  # Q2
[1] 12.5
> median(y[y<median(y)])     # Q1
[1] 9.5
> median(y[y>median(y)])     # Q3
[1] 15.5
```

However, R provides a built-in function quantiles().

```
> x = c(8,12,7,17,14,45,10,13,17,13,9,11) # raw data set
###### Quartiles ######
> quantile(x, probs = seq(0, 1, 0.25), type = 2)
  0%  25%  50%  75% 100%
 7.0  9.5 12.5 15.5 45.0
###### 42nd percentile ######
> quantile(x, probs = seq(0.42, 0.42, 0.01), type = 3)
42%
 11
```

The argument '*type*' is an integer between 1 and 9 selecting one of the nine available algorithms used to estimate quantiles.

Exercise 5.3.38 *Repeat the calculations above using different quantiles algorithms (that is, changing the values of the argument 'type' between 1 and 9). How do the results change?*

The issue here is that the definitions given above for quartiles and percentiles are **not completely satisfactory**. If we consider for instance the first quartile $Q_1 = 9.5$ calculated above for the data set of Table 5.51, one can immediately observe that there are 3 values in the data set that are smaller than Q_1, and 9 values that are larger than Q_1. It is not true therefore that 25% of the values are smaller than Q_1 and 75% are larger than Q_1. Quantiles, percentiles and quartiles, are only **approximations**, and as such can be calculated using different algorithms which may yield slightly differing results. Therefore, when reporting on a statistical study, it is important to specify clearly what algorithm was used for the computation of measures of position such as quantiles.

Exercise 5.3.39 *Write a R function to calculate the percentile rank of a given data set (that is, a vector of values).*

Exercise 5.3.40 *The Tri-City School District (Buffalo, Illinois, USA) has instituted a zero-tolerance policy for students carrying any objects that could be used as weapons. Table 5.52 give the number of students suspended during each of the past 12 weeks for violating this school policy. Determine the values of the three quartiles and the interquartile range. Where does the value of 10 fall in relation to these quartiles? Calculate the (approximate) value of the 55th percentile. Find the percentile rank of 7.*

Suspended students											
1	59	1	21	17	6	9	1	1	43	6	5

Table 5.52: Number of students suspended for violating school weapon policy.

Exercise 5.3.41 *Table 5.53 gives the numbers of new cars sold at a dealership during a 20-day period. Calculate the values of the three quartiles and the interquartile range. Where does the value of 4 lie in relation to these quartiles? Find the (approximate) value of the th percentile. Give a brief interpretation of this percentile. Find the percentile rank of 10. Give a brief interpretation of this percentile rank.*

Sold cars									
8	5	12	3	9	10	6	12	8	8
4	16	10	11	7	7	3	5	9	11

Table 5.53: Number of sold cars.

5.3.6 Box-and-Whisker plot

A **box-and-whisker plot** (sometimes simply called a box plot) gives a graphic presentation of data using **five measures**: the median, the first quartile, the third quartile, and the smallest and the largest values in the data set. These measures are called the **five-number summary**. A box-and-whisker plot can help us visualize the **center**, the **spread**, and the **skewness** of a data set. It also helps detect outliers.

In R the function fivenum() gives the five-number summary. Try it.

```
> fivenum(data_frame)
```

Let us see how to generate a box-and-whisker plot with an example. The data in Table 5.54 are the incomes in thousands of CHF for a sample of 12 households.

The following steps are performed to construct a box-and-whisker plot. First, one needs to find the five-number summary.

Income ($\times 1000\,\$$)											
75	72	84	142	74	124	81	90	99	184	79	108

Table 5.54: Incomes of 12 households.

- Rank the data in increasing order,

$$72 \quad 74 \quad 75 \quad 79 \quad 81 \quad 84 \quad 90 \quad 99 \quad 108 \quad 124 \quad 142 \quad 184\,.$$

- Calculate the **median**, which is in our case $(84 + 90)/2 = 87$.
- Calculate the first and third **quartiles**, that is, $Q_1 = (75 + 79)/2 = 77$ and $Q_3 = (108 + 124)/2 = 116$, respectively.
- The points that are $1.5 \times IQR$ below Q_1 and $1.5 \times IQR$ above Q_3 are called the **lower** and the **upper inner fences**, respectively. One can verify that the lower and upper inner fences in our case are 18.5 and 174.5, respectively. The five-number summary is then completed by finding the **smallest** and **largest** values in the dataset within the two inner fences, that is, in our case 72 and 142, respectively.
- **Note**. The R function fivenum() gives the three quartiles and maximum and minimum values of a data set, not the inner fences. The latter should be calculated using the definitions given above.

Then, one can generate the plot.

- Draw an axis such that all the values in the given data set are covered.
- Draw a **box** with its lower side at the position of the first quartile and the upper side at the position of the third quartile. Inside the box, draw a line at the position of the median.
- Join the points corresponding to the smallest and the largest values within the inner fences to the box. The two lines that join the box to these two values are called **whiskers**.
- **Any value that falls outside the two whiskers is an outlier**.

Figure 5.34 shows a box-and-whisker plot for the data of Table 5.54. The plot was generated using the following R code.

```
# Generation of a Box and Whisker plot
> df <- data.frame(Income = c(75, 72, 84, 142, 74, 124, 81,
    90, 99, 184, 79, 108))
> box <- ggplot(data=df, aes(x=factor(0),y=Income))
> box + geom_boxplot(outlier.size = 3.5) + ylab("Income") +
    stat_boxplot(geom ='errorbar',width = 0.5) + theme(text
    = element_text(size=20), axis.title.x=element_blank(),
    axis.text.x=element_blank(),
    axis.ticks.x=element_blank())
```

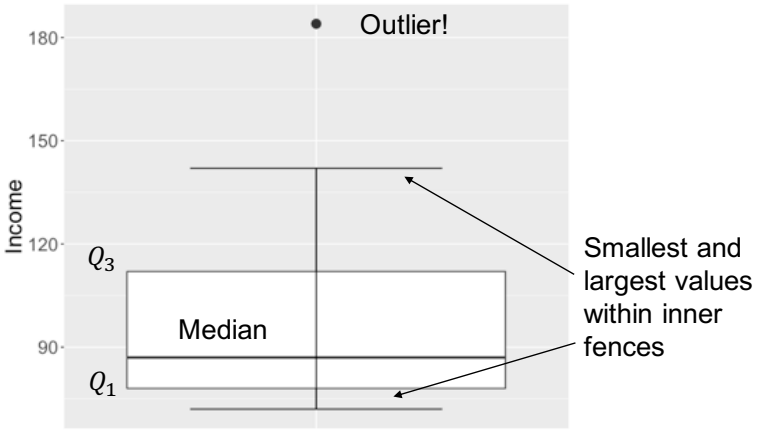


Figure 5.34: Box and Whisker plot.

The data of this example are clearly skewed (upward) because the lower 50% of the values are spread over a smaller range than the upper 50% of the values.

Exercise 5.3.42 *Table 5.55 gives the recent estimates of crude oil reserves (in billions of barrels) of Saudi Arabia, Iraq, Kuwait, Iran, United Arab Emirates, Venezuela, Russia, Libya, Nigeria, China, Mexico, and the United States. The reserves for these countries are listed in that order. Prepare a box-and-whisker plot. Are the data symmetric or skewed?*

Oil reserves (billions of barrels)					
261.7	112.0	97.7	94.4	80.3	64.0
51.2	29.8	27.0	26.8	25.0	22.5

Table 5.55: Crude oil reserves for some countries (see exercise text).

Exercise 5.3.43 *Refer again to Exercise 5.3.40. Table 5.52, showing the number of students suspended for bringing weapons to schools in the Tri-City School District for each of the past 12 weeks, is reproduced here as Table 5.56. Make a box-and-whisker plot and comment on the skewness of these data.*

Suspended students										
1	59	1	21	17	6	9	1	1	43	6	5

Table 5.56: Number of students suspended for violating school weapon policy.

Chapter 6

Project Work

The main purpose of this project-work is to apply the knowledge gained so far in the course, and also to introduce some additional elements. You are going to work with FBI crime data for the United States from the year 2013, specifically with offenses known to law enforcement categorised by various types of crimes.

6.1 Importing, First Inspection, and Cleaning

1. Go on Moodle and download the file *fbi.csv*. Import the data into R and inspect it. What are the variables? What is their type? What do they mean and how are they related to each other? Also inspect the CSV file directly in a text editor. (We strongly suggest that you use a text editor and not a spreadsheet software like excel.) Compare the variable names in the CSV and in the R data frame. You will see that the R variables do not contain white spaces, as they pose problems.

2. The data frame contains some extra columns and rows, which do not correspond to variables. Delete them. Hint: Delete the unwanted rows directly in the CSV file; doing so in R is messy.

3. The variable *Violent Crimes / Population* is not tidy. Tidy it. Note: make sure that the variables you introduce do not contain white spaces.

4. The numeric data from the CSV is interpreted as factors due to the prime character separating the thousands. Fix this. Hint: use gsub and then convert the result to a numeric type.

5. Look at the variable *State*. What do you observe? Solve the problem using a short program. It can be done with one for loop over the *State* variable, containing one if-else statement and two assignment operations.

6. The variable *Burglary* contains missing entries. Apply the *Imputation of the Mean* method and store the result in the data frame in a new variable *Burglary_iom*.

7. Apply *Regression Imputation* as described in the script and store the result in the data frame in a new variable *Burglary_reg*. Hints: Predict the variable *Burglary* from the variable *Property crime* and its subclasses, i.e. the variables *Larceny-theft*, *Motorvehicle theft*, and *Arson*. For the prediction to work, the NA entries in *Arson* have to be set to 0.

8. Compute the mean and sample variance of the two new variables. What do you observe and why?

6.2 Exploratory Analysis: Overview

1. Normalise all crime variables to crimes per inhabitant.

2. We would like to look at the variable *Violent Crimes*. First, delete the rows in the data frame, where the variable *Violent Crimes* contains an NA. If you run *df[176,]* you will see that the rowname "177" does not correspond anymore to the row index 176. This is due to the deletion. You can reassign the rownames to the row indexes with *rownames(df) = seq(1,nrow(df))*.

3. Make a graph with boxplots of the variable *Violent Crime* fanned out according to the *State* variable, i.e. where each boxplot in the graph corresponds to a state. Which state has the most extreme outliers (i.e. greater than 0.6)? Do you have an explanation for it? Hint: Use the *which* function to find the index of the maximum entry and look at the row.

4. Many states have extreme outliers. One way to "zoom" into the graph is to set the range of the y-axis. Try with values < 0.1 until you see the structure of the boxplots.

5. What is your expectation for the skewness of *Violent Crime*? Compute it.

6. Compute the skewness for each state. Hint: Use the function *tapply* with the function *skewness*. One of the states has NA as skewness. Which one and why?

7. Find the *State* with highest and lowest skewness. Hint: Use the *which* function together with *max(..., na.rm = FALSE)*.

6.3 Exploratory Analysis: Detailed View

1. In the previous section you should have found COLORADO to have the most extreme outlier, and CALIFORNIA and NEVADA to have the max and min skewness. Construct a new data frame using the command

```
df_caconv = df[which(df$State=="CALIFORNIA" |
    df$State=="COLORADO" | df$State=="NEVADA"), ]
```

Re-use your code for the boxplots above to show a boxplot for each of the three states in one graph. Also try y-ranges smaller than 0.1.

2. Create a dataframe for CALIFORNIA only, and make a histogram for the variable *Violent Crime*. Due to the outliers most of the values land in the first bins only. One way to obtain a better histogram would be to exclude the outliers. Alternatively we can put them all in one bin. Try to understand the following code and apply it (assuming *df_ca* is the dataframe you just created):

```
ggplot(df_ca, aes(x = df_ca$Violent.crime)) +
  geom_histogram(breaks=c(seq(0,0.05,0.001),
      max(df_ca$Violent.crime)),
                 color="blue", fill="steelblue")
```

3. Create a dataframe for NEVADA only. Make a scatterplot of the variables *Property Crime* and *Violent Crime* and a add a trendline, once using default parameters and once using a linear fit. Hints: Use the *geom_smooth* and find out how to tell the function to use a linear fit.

4. Do you think there is a correlation between emphProperty Crime and *Violent Crime*? Let us look at all pairs of numerical variables and create a heatmap of their correlations. Try to interpret the heatmap. Hints: Produce a dataframe with numerical variables only and feed it to the function *cor*, which returns a matrix. Use *geom_tile* to plot the heatmap. To do so, you need to convert the correlation matrix to a dataframe using the function *reshape2::melt()*.

6.4 Frequency Table

For this exercises we will use the full data, i.e. with all the states.

1. Make a frequency table (as described in the script) for the variable *Property Crime* using 50 classes to group the data. We have seen that the data is skewed, in particular, due to outliers. To remove them, exclude the 10 largest values prior to computing the frequency table.

2. Make a barplot of the frequencies.

6.5 Standard Deviation Plot

Standard deviation plots are used to see if the standard deviation varies between different groups of the data. Although the standard deviation is the most commonly used measure of dispersion, the same concept applies to other measures of dispersion. Certain model based statistical tests assume the variances between the groups to be equal. Standard deviation plots are way to test this

assumption visually (though there are also statistical tests to do so, which is outside the scope of this course). The following code provides one way to obtain a standard deviation plot. Try to understand and run it (assuming *df* is the full dataframe, containing all states):

```
df_var_state <- as.data.frame(tapply(df$Property.crime,
    df$State,FUN=sd), row.names = NULL)
colnames(df_var_state) <- "Std"
df_var_state$State <- rownames(df_var_state)

ggplot(data=df_var_state, aes(x=State, y=Std)) +
  geom_point(aes(size=0.3)) +
  theme(text = element_text(size=11),
      axis.text.x=element_text(size=10,angle=90, hjust=1)) +
  theme(legend.position="none")
```

Add a line of code to remove COLORADO from the analysis.

6.6 Small Programming Tasks

As programming exercise, in the following two tasks you will re-implement functions that already exist in R. They take one numerical vector as argument, return a number, and should not make use of the corresponding R function (e.g. do not simply call *max* if you are asked to implement your own *mymax* function).

1. Write a function *mymax* that returns the largest value in the vector.
2. Write your own skewness function. It should return the sample skewness, using the method of moments as described on Wikipedia:
   https://en.wikipedia.org/wiki/Skewness
   Test your function, by comparing its output to *e1071::skewness*.

Optional: Write a function that takes a numerical vector as argument and prints a stem-and-leaf display.

Bibliography

[1] Stef Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

[2] Tamraparni Dasu and Theodore Johnson. *Exploratory data mining and data cleaning*, volume 479. John Wiley & Sons, 2003.

[3] Edwin de Jonge and Mark van der Loo. An introduction to data cleaning with r. *Statistics Netherlands, The Hague*, page 53, 2013. Available at http://cran.stat.nus.edu.sg/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf.

[4] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2014.

[5] Prem S. Mann. *Introductory Statistics, Seventh Edition*. John Wiley & Sons, 2009.

[6] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007. OBO ontologies are available at http://www.obofoundry.org/.

[7] Nikos Tsikriktsis. A review of techniques for treating missing data in om survey research. *Journal of Operations Management*, 24(1):53–62, 2005.

[8] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer Science & Business Media, 2009.

[9] Hadley Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.

[10] Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(1):1–23, 2014. Available at https://www.jstatsoft.org/index.php/jss/article/view/v059i10.

[11] Hadley Wickham and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, 2016. Available at http://r4ds.had.co.nz/.

[12] Leland Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.