FOUNDATIONS
OF
DATA
SCIENCE

Foundations of Data Science (Inf2-FDS) 2023
Lecture Notes

David Sterratt and Kobi Gal
with some material based on notes for Inf2B by Hiroshi Shimodaira, Iain Murray and Steve Renals

26th November 2023

Contents

About these lecture notes

These lecture notes are written for the University of Edinburgh Course Informatics 2 – Foundations of Data Science. Ultimately we aim that they will give comprehensive (though perhaps not exhaustive) coverage of the *material* covered in the lectures.

Although the notes are designed to fit in with the Foundations of Data Science course, we're making them an open educational resource available under a Creative Commons licence. The development of this course has benefited from other open educational resources, and now the notes are fairly well developed (though still not complete or error free!) it seems like a good time to give back to the world.

How to use these notes

The process of learning involves more than just reading the material, and we strongly encourage students on Foundations of Data Science to take part in the learning activities that are designed to go with these notes: the lectures themselves, "comprehension questions", computer labs, tasks and workshops, and the coursework. You should look at the course website for details of all of these activities. In these notes we've linked to the labs, which are also openly available.

These notes are still a work in progress; some chapters are more developed than others, with some still being outlines. In previous years we have shared only the mature chapters, but this year (2023–24) we're sharing the whole set of notes, partly to give students a better overview of the course, and partly to encourage us to improve the completeness of the notes. There are doubtless mistakes in the notes, and we welcome reports of errors and suggestions for improvements.

The philosophy of Foundations of Data Science

It is often said that 80% of the time spent on a data science project is in preparing the data, and the quality of the data is crucial to the outcome of any data science or machine learning project. The philosophy of Foundations of Data Science is to allow students to acquire the foundational skills of data, before moving on to other courses (such as Machine Learning), which build on these foundations.

As stated in the Learning Outcomes, we intend that on completion of this course, the student will be able to:

- Describe and apply good practices for storing, manipulating, summarising, and visualising data.
- Use standard packages and tools for data analysis and describing this analysis, such as Python and LaTeX.
- Apply basic techniques from descriptive and inferential statistics and machine learning; interpret and describe the output from such analyses.
- Critically evaluate data-driven methods and claims from case studies, in order to identify and discuss a) potential ethical issues and b) the extent to which stated conclusions are warranted given evidence provided.
- Complete a data science project and write a report describing the question, methods, and results.

Although we present (fairly informally) the mathematical foundations of methods such as linear regression, the focus of FDS is more on understanding the principles and the correct application of methods rather than the derivation or development of methods.

Much of data science is about story telling and explanation, so we consider that models such as linear regression are helpful not only for prediction, but also for explanation. We therefore cover topics such as the interpretation of linear and logistic regression coefficients.

We first approach statistical inference from the point of statistical simulations and sampling theory, inspired by the computational approach to statistical inference in the Berkeley Data 8 course, and its associated textbook. Given the ubiquity of using theoretical distributions to generate confidence intervals and undertake hypothesis testing, we also introduce some common statistical distributions and tests.

The structure of the notes

We have organised these notes into 6 separate parts, each containing multiple chapters:

1. Data wrangling and exploratory data analysis
2. Introduction to Machine Learning
3. Linear models
4. Statistical inference
5. Regression and inference
6. Project skills

This is a slightly different way to the structure envisaged in the FDS course descriptor, but all the topics indicated there are included. Interleaved with these topics are topics focusing on real-world implications (often using case studies), critical thinking, working and writing skills.

The order of the chapters represents one ideal ordering of the material. For logistical reasons the order of the lectures differs from this ideal order: We want to present the material on statistical inference early enough to be useful for a coursework at the beginning of Semester 2, but late enough so that students will have covered the relevant concepts of probability in Discrete Maths and Probability.

Influences

Material on supervised learning and clustering mostly comes from the work of Steve Renals, Iain Murray and Hiroshi Shimodaira on the previous Informatics course Inf2B.

We are indebted to other openly-available data science courses, in particular Harvard's CS109, in particular the concept of the data science life cycle, and the Berkeley Data 8 course, with its emphasis on programming statistical simulations for inference.

Gelman and Nolan's (2017) book *Teaching statistics – a bag of tricks* has provided the basis for presentation of some of the concepts and lecture demos.

Resources

There is no one textbook that covers all this course. However, we will refer to the following books and resources at points throughout the course:

***Modern Mathematical Statistics with Applications* (Devore and Berk, 2012):** *Modern Mathematical Statistics with Applications* is an introduction to inferential statistics that follows on from the Discrete Maths and Probability course. It's freely available as a PDF from the library - see the Library Resources link. You can also purchase a print copy for £25 via the page for the book that you reach via the Library.

***The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios* (Wexler et al., 2017)** The first chapter of this book provides a concise introduction to principles of visualisation.

***An Introduction to Data Ethics* (Vallor, 2018)** This is an introduction to data ethics from the perspective of *virtue ethics*. It presents case studies to help you develop your ethical sensitivity and ethical reasoning.

***Computational and Inferential Thinking* (Adhikari et al., 2020):** The online textbook *Computational and Inferential Thinking* takes a computational, non-mathematical approach to statistical inference, and the statistical inference sections of FDS have drawn inspiration from their approach. Please note that Python code in the book uses a bespoke python library instead of pandas, which we'll be using on this course, so don't pay too much attention to the details of the code.

***Becoming a critical thinker: for your university studies and beyond* (Ivory, 2021)** This book, available online in the University Library, is an excellent general introduction to critical thinking. We recommend the whole book, but if you have little time, read Chapters 4–6 (on arguments, evidence and communication). This reading will help you to understand the sort of thinking, writing and referencing that we would like to see in the Critical Evaluation and the Project.

Notation

The following is the default notation used in the course. Note that different versions may be used sometimes for the ease of readability.

x	A scalar
$\mathbf{x}$	A column vector : $\mathbf{x} = (x_1, x_2, \ldots, x_D)^T$, where D is the dimension of vector
$\mathbf{x}^T$	Transpose of vector $\mathbf{x}$, meaning a row vector if $\mathbf{x}$ is a column vector
x_i or x_d	ith sample (scalar), or dth element of vector $\mathbf{x}$
$\sum_{i=1}^N x_i$	Summation, i.e., $x_1 + x_2 + \ldots + x_N$
$\prod_{i=1}^N x_i$	Product, i.e., $x_1 x_2 \cdots x_N$
$\mathbf{x}_i$	ith sample (vector): $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$
$\|\mathbf{x}\|$	Euclidean norm or L^2 norm: $\|\mathbf{x}\| = \sqrt{\sum_{d=1}^D x_d^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$, also known as magnitude
$\|\mathbf{x}\|_1$	L^1 norm, i.e. $\sum_{d=1}^D \|x_d\|$
$\mathbf{u} \cdot \mathbf{v}$	dot (inner or scalar) product of $\mathbf{u}$ and $\mathbf{v}$, i.e., $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \, \|\mathbf{v}\| \cos\vartheta = \mathbf{u}^T\mathbf{v} = \Sigma_{d=1}^D u_d v_d$
A	A matrix: $\mathsf{A} = (a_{ij})$. If A is a M-by-N matrix, $\mathsf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$
A_{ij}	Element of matrix A at ith row and jth column, i.e. a_{ij}
$\mathbf{I}$ or $\mathbf{I}_d$	Identity matrix of size d by d (ones on diagonal, zeros off)
A^T	Transpose of matrix A, i.e. $\mathsf{A}^T = (a_{ji})$
A^{-1}	Inverse of matrix A, i.e. $\mathsf{A}^{-1}\mathsf{A} = \mathsf{A}\mathsf{A}^{-1} = \mathbf{I}$
$\{\mathbf{x}_i\}_1^n$	A set of samples: $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
$\bar{x}$	Sample mean of x
s^2, s_x^2, or $\mathrm{Var}\,(x)$	Sample variance of x
s_{xy}	Sample covariance of x and y. NB: $s_{xx} = s_x^2$
μ	Population mean
σ^2	Population variance
$\boldsymbol{\Sigma}$	Population covariance matrix (cf. summation operator $\sum$)
σ_{ij}	(i, j)-element of a covariance matrix, i.e. $\boldsymbol{\Sigma} = (\sigma_{ij})$. NB: $\sigma_{ii} = \sigma_i^2$
$\exp(x)$	The natural exponential function of x, i.e. e^x
$\ln(x)$	The natural logarithm of x
$x \propto y$	x is proportional to y

This notation is also used by *Modern Mathematical Statistics with Applications* (Devore and Berk, 2012).

Part I

Data: ethics, collection, representation, wrangling, exploration, visualisation and descriptive statistics

Chapter 1

Introduction and Logistics

1.1 What is data science?

Data science is all about exploring, explaining, and inferring insights from vast amounts of data. The end-goal of this course is to provide students with tools they require to be great data scientists, the lucky practitioners that develop and/or use data-scientific technologies! So let's first define what a data scientist does.

What is a data scientist? Data scientists are able to turn raw data into understanding, insight and knowledge. Some of the activities that data scientists do on a daily basis include the following:

- Find, collect, check and clean data.
- Explore data and infer knowledge and insights.
- Explain and interpret these inferences.
- Communicate inferences about the data to others.
- Make prediction about future trends.

This is why data scientists need to employ interdisciplinary tools, from statistics, artificial intelligence and machine learning. We will be addressing all of these different aspects in the course, and have closely aligned our learning goals with the varied skill set that data scientists need to be successful in the field.

This process is broken up into steps and shown in Figure 1.1. We will expand on all the steps of the data science cycle in this course.

The data science process We can see data science as a process, comprising interconnected steps (Figure 1.1).

The data explosion: volume, variety and velocity Data is always around us, and it always has been. But the development of technology has made it possible to collect and store vast amounts of data. Just to get some perspective, each day on Earth we generate over 500 million tweets (how many of these are generated by bots?), 294 billion emails, 4 million gigabytes of Facebook data, 65 billion WhatsApp messages and 720,000 hours of new content added daily on YouTube.

In 2018, the total amount of data created, captured, copied and consumed in the world was 33 zettabytes (ZB) – the equivalent of 33 trillion gigabytes. This grew to 59ZB in 2020 and is predicted to reach a mind-boggling 175ZB by 2025. One zettabyte is 8,000,000,000,000,000,000,000 bits (Vopson, 2021).

Parallel to this explosion of data generation, there is also consistent improvement in computational methods; we are now able to mechanise some aspects of data analysis that data scientists care about. So, while the amount of digital data in the world doubles every two years, so does the computational power that lies at our disposal to analyse it. One should state that the speed of computation, which was commonly considered to double every two years, may have plateaued.

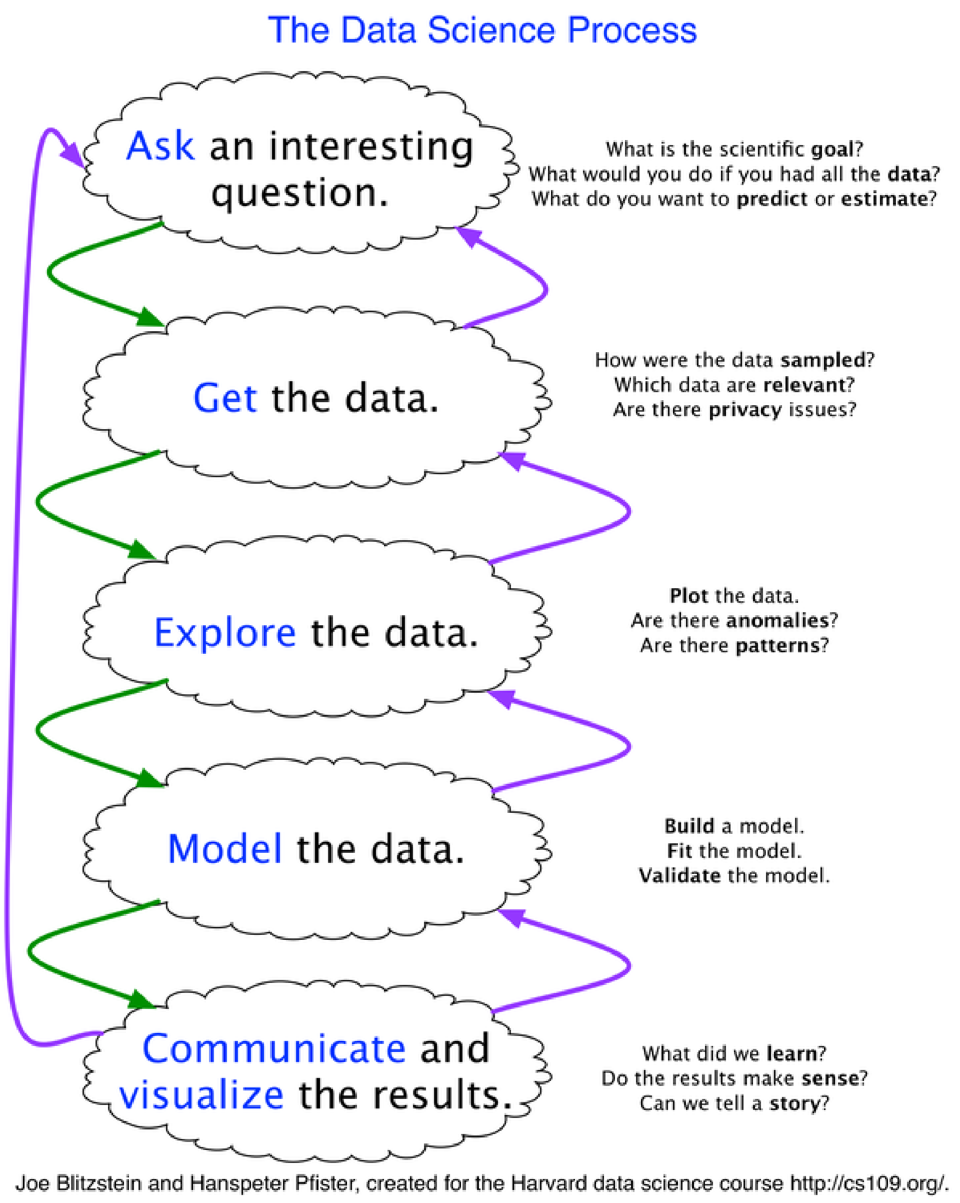

The Data Science Process
Ask an interesting question.
What is the scientific goal?
What would you do if you had all the data?
What do you want to predict or estimate?
Get the data.
How were the data sampled?
Which data are relevant?
Are there privacy issues?
Explore the data.
Plot the data.
Are there anomalies?
Are there patterns?
Model the data.
Build a model.
Fit the model.
Validate the model.
Communicate and visualize the results.
What did we learn?
Do the results make sense?
Can we tell a story?
Joe Blitzstein and Hanspeter Pfister, created for the Harvard data science course http://cs109.org/.

Figure 1.1: The data science process. Credit: Joe Blitzstein and Hanspeter Pfister, created for the Harvard data science course CS109. This figure is not under the CC BY-SA licence covering the rest of the notes.

How does data science relate to other fields? Data Science, Statistics, Machine Learning. These are some of the terms used in the scientific literature and popular media. Let's try to sort through the confusion.

Data science is a multidisciplinary field which uses scientific methods, processes, and systems in a range of forms. Both statisticians and data scientists care about analysing and explaining data. Indeed, some of the methods used by data scientists are taken from statistics, and we shall learn them in this course.

Statisticians focus on using structured models and parameter estimation to quantify uncertainty in data. Data science uses methods from statistics and machine learning. but in addition, data scientists need to do a lot of work on processing the data itself and check that the data makes sense. Data scientists often deal with huge databases, which is why they rely heavily on computational methods for their analysis.

Machine learning is all about algorithms that are able to learn from data. Data science also uses some tools from machine learning, and we will study some of these in the course. Data scientists begin by exploring the data and formalise questions that can be asked on the data. They care about explaining how (and possibly why) trends in the data arise. They need to communicate quantitative and qualitative arguments to the general public. Often, they also care about supporting practitioners in the field (e.g., alerting teachers to struggling students based on their interactions with educational software). The value of a machine learning algorithm is measured by its performance on unseen data. But the value of a data science analysis also needs to consider the human in the loop. We're (happily) not at the stage when these complex tasks can be replaced with computer algorithms. There is a lot of room for skill and creativity that cannot be automated.

In all these fields, it's important to know where the data comes from, how it's been collected, and to be able to reason about whether a dataset has been collected ethically, or if the project we are undertaking is ethical.

Examples of data science at work Let's consider two examples of data science at work. Both of these examples are meant to highlight the inherent biases that unfortunately exist in society, and that also arise in the data sets that we generate.

- Example of gender tropes in films. In this example we will see the gender bias reflected in script writing in movies.
- Case study: COMPAS. In this example we will describe racial bias in the criminal justice system.

Data science career prospects Data science is a lucrative field. Although the study and analysis of data started off in academia, industry is increasingly taking a leading role and applying and developing data science methods. Demand is high for data professionals—data scientists and mathematical science occupations are growing at a significant higher rate than in other high tech fields.

1.2 Course logistics

- Learning outcomes
- Course activities
- Assessment

Chapter 2

Data

2.1 Data and metadata

What is data? Since the 17th Century, the word **data** has referred to "things known or assumed as facts, and made the basis of reasoning or calculation" (OED). Data according to this definition has existed for 1000s of years, for example in the form of handwritten tables of scientific measurement, census records and financial accounts such as those found in Ancient Egypt (Figure 2.1). This data is all collected by humans, and is typically written down. The amount of data it is possible to collect is therefore low, and any calculations performed on the data had to be done by hand.

Since the mid-20th century, the word has also referred to "the quantities, characters or symbols on which operations are performed by computers and other automatic equipment, and which may be stored and transmitted in the form of electrical signals, records or magnetic, optical or mechanical recording media" (OED). Thus, the definition of data has expanded. Digital data comes in many forms, for example images, sound files, emails and documents, scientific databases and medical records. Some of these forms of data are collected by machine or automatically – for example, a digital camera sets creates the set of numbers that describe an image; a web server records the details of the every visit to a web page. The amount of data it has been possible to collect has increased massively, as has our ability to process that data.

Nevertheless, in this age of big data, humans are still collecting smaller datasets. For example, opinion pollsters might survey 1000 people in a poll, or ecologists might measure the height and weight of a sample of 100 squirrels.

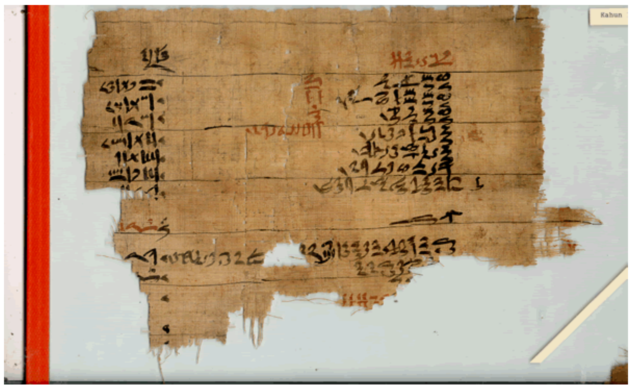


Figure 2.1: Ancient Egyptian grain account from between 2025–1700 BC found in the Lahun Papyri. Acquisition UC32189, Digital Egypt for Universities, University College London, `https://www.ucl.ac.uk/museums-static/digitalegypt/lahun/papyri.html`. Courtesy of the Petrie Museum of Egyptian and Sudanese Archaeology, UCL.

Table 2.1: Characteristics of some imaginary squirrels.

Name	Weight (g)	Length (mm)	Sex	Age
Jakub	320	211.0	Male	Under 1 year
Fiona	342	222.0	Female	1–2 years
Cameron	330	215.0	Male	2+ years

Structured and unstructured data There is an important distinction is between structured data and **unstructured data**. In **structured data**, the data is organised according to a clear structure. An example of structured data is a table describing attributes of Star Wars characters with columns such as name, "height", "weight" and eye colour. Structured data needn't be a table: for example a bibliographic database, with different types of entries for books and academic articles. It is easy to query structured data: for example, find all characters in Star Wars with blue eyes.

Unstructured data does not have a predefined structure. Text and images are common examples of unstructured data. It is typically much harder to extract information from unstructured data than from structured data. For example, in a block of text written about Star Wars characters, would find it hard to identify all characters with blue eyes.

Metadata We contain terabytes of data files, but they are useless if we know what they *mean*. **Metadata** – literally "about data" – is information describing the data files it accompanies. The metadata may be a description in a text file (often called README) or it may be in a structured format. Whenever a dataset is created, stored and shared, the data meaning of the data should be recorded. It is surprising how often the meaning of datasets is not clearly described. The metadata should describe how the data were collected, including if there were any legal or ethical considerations, the format of the data files and the meaning of variables in the data files.

2.2 Tabular data and variables

Tabular data In this course, we focus on a common class of structured data, **tabular data**. Tabular data is data arranged in a table. The meaning of each cell in the table is determined by the column and/or row labels. Table 2.1 shows an example of tabular data.

Tidy Data **Tidy data** is a subset of tabular data (Wickham, 2014). In a tidy dataset, each column corresponds to a **variable** (or **attribute**) and each row corresponds to an **instance** or **observation** possessing each of the variables. The terms **data matrix** or **long form data** are synonyms for tidy data.

For example, suppose we capture squirrels and measure their weight (in grams) and length (in millimetres), and note down their sex. The variables represented in the columns would be "Name" (assuming we've named the squirrels we are observing), "Weight (g)", "Length (mm)" and "Sex". Each row contains the value of these variables.

Typically, tidy data is **multivariate**, i.e. it has multiple variables. In the special case of two variables we refer to it as **bivariate**, and when we consider only one variable, we call it **univariate**.

Messy data We call data that is not in the tidy format messy. There are a number of ways in which data can be messy, for example:

- The values of variables could be column headings, as shown in Table 2.2. Here the messy format is easy for humans to read, but is often more difficult to process. Functions such as `melt` in Pandas can be used to rearrange this messy format into a tidy format.
- A single observational unit could be stored in multiple tables. For example, data on the CO2 emissions for countries over time is stored in per-continent files, e.g. `annual_co2_europe.csv`, `annual_co2_asia.csv`... To tidy the data here, we would load in each file, give it a column "Continent" with the value for each row derived from the file name, and then concatenate these files.

Table 2.2: Messy data with values as columns. In the messy table (top), the rows correspond to one variable ("Sex") and the columns "Under 1 year old", "1 to 2 years old" and "More than 2 years old" are actually the values of "Age", a categorical variable. This format is easy to read, but it is not tidy, as the values of a variable are in the column names. We can rearrange to a tidy format (bottom) in which there is one column for each of the variables "Sex" and "Age", and a column "Count".

Messy data

Sex	Under 1 year old	1 to 2 years old	More than 2 years old
Male	4	7	2
Female	6	9	1

Tidied data

Sex	Age	Count
Male	Under 1 year old	4
Female	Under 1 year old	6
Male	1 to 2 years old	7
Female	1 to 2 years old	9
Male	More than 2 years old	2
Female	More than 2 years old	1

Messy data may actually be easier for the human eye to comprehend, but is not so easy to manipulate using software tools, so for the moment we will assume we have tidy data.

Types of variables Variables can be of various types:

- **Numerical variables** are quantities that can be measured (continuous) or counted (discrete). For example, weight and length are continuous numeric variables. In contrast, the number of babies a squirrel gives birth to in a year is a discrete variable, since we cannot have a fractional number of babies. Continuous variables often have a physical dimension (e.g. weight or length) and it is important to quote them with their unit.

- **Categorical variables** can take on one of a number of values. For example, the sex of the squirrel is a categorical variable, since it can be "Male" or "Female". We can have more than two categories, for example a fruit might be "Apple", "Orange", "Lemon", "Grapefruit" etc.

- **Ordinal variables** can take on one of a number of categories, but those categories are ordered. For example, we might estimate the age of a squirrel as "Under 1 year old", "1 to 2 years old" or "More than 2 years old". There are only three categories, but we can order them from youngest to oldest.

- **String variables** contain string information such as names or a comment from a survey form.

Representing categorical variables The human-readable format of categorical variables is as a string. However, for many of the algorithms that we deal with later, the string representation is not helpful. We could represent the category by a number, e.g. 0 for "Apple", 1 for "Orange", 2 for "Lemon" etc. But this system implies an ordering for the categories, which is not the case here.

Instead, what we do is convert the categorical variable into an **indicator variable**, also known as a dummy variable. We create one new column for each category (for example, an "Apple" column, an "Orange" column and a "Lemon" column). We indicate which category the item belongs to by putting a 1 in the corresponding columns, and zeros in the other columns. This form of encoding of categories is also known as **"one-hot" encoding**, since there is always 1 "hot" bit required to indicate the category.

In fact, if we have k categories, we can manage with $k-1$ columns, by dropping one column, which we regard as the default. For example, if we dropped the "Apple" column, when there were zeros in all the remaining columns, we would assume that the item represented was an "Apple".

2.3 Working with tabular data

Reading stored data Tabular data can be stored in a variety of formats, and Data Science packages (for example Pandas or R) have functions to read in this data:

- Text file: common formats are comma-separated variable (CSV), tab separated variable (TSV). There are various standards for the precise formatting of the files, for example if the data enclosed by cells are enclosed by quotes. Sometimes there are a few lines of metadata at the top of the file. Data science packages functions to read in text files, have many options, for example allowing you to skip lines at the head of the file, or deal with separators other than tabs or commas.
- Binary file: common formats are the Excel spreadsheets or Open Document Format spreadsheets. The modern versions of both of these are in fact ZIP files containing an XML file with the spreadsheet information, and any other files (e.g. embedded images).
- Databases: for example MySQL or SQLlite. Here there is a database server, and data is extracted by running SQL (Structured Query Language).[1]

Text files have the advantage of being human-readable, and also enforce a discipline of encoding information solely in the content of the cells – it is not possible to encode information by colour-coding cells, as it is in spreadsheets. Conversely, spreadsheets have the advantage be being format-able, which can help with readability.

Selecting rows and columns Data science packages have methods for extracting rows and columns from tables. With tidy data, extracting a row selects all the data connected with one observation. Extracting a column selects every instance of one variable.

Filtering data We call finding a subset of the data based on the value of some variable **filtering**. For example, we may wish to filter out all the squirrels longer than 220 mm.

2.4 Data wrangling

Before you can undertake data science analyses, you need to get (or "wrangle") the data into a suitable form, a process that is called **data wrangling**.

Cleaning data The quality of any analysis of data can only be as good as the data itself – and the data itself may have many types of problems:

- Data entry: for example, if we have collected data in a free-text survey, respondents may not have entered a time in a uniform format (e.g. "16:00", or "4pm", or "4" or "16" or "17 (CET)") may all mean the same thing. Or someone may have typed "08" when they meant "80".
- Mixing text and numbers: for example, we might have recorded "210g" or "0.21kg" in the weight cell for our squirrels.
- Missing data: perhaps a sensor wasn't working for a few minutes or hours, so some readings are missing. Does it record "0" in this situation? Or "−1". The metadata should tell us, but maybe it doesn't...
- Mislabelled data: A column may have been mislabelled. For example, we might be recording the temperature of heating water entering and leaving the Informatics Forum, and the temperature of heating water entering and leaving Dugald Stewart Building (DSB). To get a measure of how much heat the Forum is using, we subtract the temperature of the water leaving from the water entering, and the same for DSB. But if we subtract the temperature of water leaving DSB from water entering the Forum, we will get nonsense. (This scenario actually happened.)

[1]You can learn how to process and analyse data using SQL in the 3rd year course Introduction to Databases.

Table 2.3: Time taken to complete obstacle course by squirrels

Name	Date	Time (s)
Fiona	2021-04-06	67.5
Fiona	2021-04-10	50.2
Cameron	2021-04-08	55.6
Lily	2022-07-13	45.0

- Duplicated data: perhaps someone has made a copy-and-paste error in a spreadsheet.
- Faulty sensor: perhaps a sensor goes wrong, and starts giving values that are implausible.

In summary, there are many potential problems with data: **the data is out to get you!** One of the most important jobs of a data scientist is to check the data quality, and fix problems. You need to approach the data in the spirit of critical evaluation: Is this value reasonable? Is that pattern strange? Does the data look too good?

We use the term **data cleaning** to describe the process of checking and fixing problems in data. You should start data checking and cleaning after loading the dataset, but problems with data also emerge in the process of visualisation, which we will come to later.

Data cleaning is very time-consuming: it is commonly said that 80% of the time of a data science project is spent on cleaning (Dasu and Johnson, 2003) though some more recent estimates suggest time spent cleaning is less than 30% (Anaconda, 2020), though the same report also estimates loading and visualising data take around 20%. In any event, data cleaning can take a long time, needs to be done carefully, and can be quite fiddly and frustrating.

Missing data Modern data science packages have a special values – `NA` or `NaN` (**Not applicable** or **Not a Number**) – to describe data that is missing. It's very helpful to ensure that data that you regard as missing shows as `NA` or `NaN` rather than as a default value (e.g. 0 or -1), as this can help with filtering out missing data.

Pandas is somewhat confusing in its handling of missing data. Strictly speaking, `NA` applies to missing string, categorical or numeric data, whereas `NaN` applies only to missing numbers. However, Pandas represents missing data as `NaN`, but functions to check for missing data refer to `NA`, e.g. `.isna()`.

Once missing data is identified, you have to decide what to do with it, which may depend on the analysis you are undertaking. Sometimes it can be possible to keep observations that contain missing data, but some analyses only work if every variable in every observation has a value. In this case, you may need to drop any rows containing `NaN`.

2.5 Merging tabular data using database-style joins

Often we wish to combine data from two sources that relates to the same entities. For example, we might have recorded the times that some of our squirrels could undertake an obstacle course. Now, suppose we wanted to compare the times of completion to the characteristics of the squirrel (Weight, Length, Sex and Age). Both Table 2.1 and Table 2.3 share a common variable: "Name". We can match up the rows of both tables by using the `merge` function in Pandas with "Name" as the **key** that binds the tables. In relational database terminology the equivalent operation is called a **join**.

Notice that not all the squirrels in the Table 2.1 undertook the obstacle course, and there is one squirrel ("Lily") who is present in Table 2.3 but not in the first table. There are various ways of merging to deal with this mismatch, described in the next paragraphs.

Inner join In an **inner join** (Table 2.4), only the squirrels in both datasets will be present in the joined dataset. Note that the key Fiona in Table 2.1 matches two rows in Table 2.3, so there are two corresponding rows in the joined table. The data describing Fiona's characteristics is repeated in these rows, but the unique information about the data and time of the obstacle course run are not repeated. Jakub isn't present in this table, since Jakub hasn't had a time recorded on the obstacle course in Table 2.3.

Table 2.4: Results of inner join applied to Tables 2.1 and 2.3.

Name	Weight (g)	Length (mm)	Sex	Age	Date	Time (s)
Fiona	342	222.0	Female	1-2 years	2021-05-06	67.5
Fiona	342	222.0	Female	1-2 years	2021-05-10	50.2
Cameron	330	215.0	Male	2+ years	2021-05-08	55.6

Table 2.5: Results of left join applied to Tables 2.1 and 2.3.

Name	Weight (g)	Length (mm)	Sex	Age	Date	Time (s)
Jakub	320	211.0	Male	Under 1 year	nan	nan
Fiona	342	222.0	Female	1-2 years	2021-05-06	67.5
Fiona	342	222.0	Female	1-2 years	2021-05-10	50.2
Cameron	330	215.0	Male	2+ years	2021-05-08	55.6

Left and right joins In a **left join** (Table 2.5), all squirrels present in the first (left) table passed to the merge function will be retained in the merged table. When the key isn't present in the second table, we fill in the missing values with `NaN`. In Table 2.5 Jakub has `NaN` values for data and time, since Jakub hasn't had a time recorded on the obstacle course in Table 2.3.

In a **right join** all items in the right-hand table are retained, and the resulting table will have `NaN` values when a key is present in the right-hand table but not the left-hand table.

Table 2.6: Results of outer join applied to Tables 2.1 and 2.3.

Name	Weight (g)	Length (mm)	Sex	Age	Date	Time (s)
Jakub	320.0	211.0	Male	Under 1 year	nan	nan
Fiona	342.0	222.0	Female	1-2 years	2021-05-06	67.5
Fiona	342.0	222.0	Female	1-2 years	2021-05-10	50.2
Cameron	330.0	215.0	Male	2+ years	2021-05-08	55.6
Lily	nan	nan	nan	nan	2022-07-13	45.0

Outer join In an **outer join** (Table 2.6): All squirrels in both datasets will be present in the joined dataset. When the key isn't present in the first or second table, we fill in the missing values with `NaN`. In Table 2.6 Jakub and Lily are present because they are present either in Table 2.1 (Jakub) or Table 2.3 (Lily). Jakub has `NaN` values for data and time, since Jakub hasn't had a time recorded on the obstacle course, and Lily's age, height, weight etc. are missing, since Lily didn't appear in Table 2.1.

2.6 Split-apply-combine

Split-apply-combine A commonly used operation in data science is **split-apply-combine**. It involves:

- Splitting the data into groups based on some criteria.
- Applying a function to each group independently. This could include computing a summary statistic for each group, such as a mean or count; performing some group-specific computation such as standardising the data within a group; as well as filtering to discard data that has only a few members, for example.
- Combining the results into a data structure.

For example, we might want to find the fastest time for male and female squirrels. In this case, we could split the joined table (Table 2.4) by Sex, apply the `min` function to each subtable, and then combine the results in one table (Table 2.7).

Table 2.7: Results of **splitting** by Sex, **applying** `min()` and then **combining**, applied to the inner join (Table 2.4).

Sex	Time (s)
Female	50.2
Male	55.6

Advantages of split-apply-combine Data science programming environments, such as R or the Pandas package in Python have functions that can achieve split-apply-combine operations in one line. We could write code to achieve the same effect, but it would be less clear, at least to data scientists who are familiar with data science programming idioms.

The split-combine-apply paradigm is also found in big data, where it is referred to as MapReduce. Here the groups created by the split operation can be processed in parallel before being combined.

Related Python Lab: Introduction to Jupyter notebooks and Pandas

`https://github.com/Inf2-FDS/FDS-S1-01-introduction`

This lab covers

- Jupyter notebooks: cell types, running cells and shortcuts
- Pandas
  - Series
  - DataFrames
  - Reading CSV files into DataFrames, and observing properties of DataFrames
  - Selecting rows and columns
  - Filtering

We also share the Inf2-IADS Lab 1, which covers:

- The Python interpreter
- Basic data types: Numbers (int and float), strings, booleans, lists (including list comprehension), tuples, sets and dicts.

Related Python Lab: Pandas – Data wrangling

`https://github.com/Inf2-FDS/FDS-S1-02-data-wrangling`

- High level data exploration: `.info()` and `.describe()`
- Cleaning data
  - Detecting non-numerical values
  - Dealing with non-numerical values: dropping or overwriting
  - Finding unique or duplicated values
- Combining data from two datasets: concatenation and merging

Related Python Lab: Data wrangling II

`https://github.com/Inf2-FDS/FDS-S1-05-data-wrangling-02`

- **regular expressions** (regexps)
- dummy variables, indicator variables
- Simple web scraping with `pd.read_html`
- Split-apply-combine (groupby)

Chapter 3

Descriptive statistics

Recommended reading

Modern Mathematical Statistics with Applications, Sections 1.1, 1.3 and 1.4

3.1 Introduction to descriptive statistics

Statistics The German word *Statistik* arose in the 18th century and originally referred to “data about the *state*” (country). The first use of “statistical” in the English language was in 1791 in the *Statistical Account of Scotland*. Sir John Sinclair, an elder in the Church of Scotland, sent a questionnaire to ministers in every parish (church district) in Scotland. The questionnaire asked many questions about agriculture, industry, economics, employment, poverty and education, as well as “The state of the manners, the morals, and the religious principles of the people”. In fact empires and dynasties have been collecting data about population and trade for much longer than this, going back to the Han dynasty in China and the Roman Empire.

Descriptive statistics It’s not humanly possible to make sense of a large raw datasets. For example, suppose we know the salary of every member of staff in the University of Edinburgh – the list would be very long. To make sense of this data we can try to summarise it or visualise it. **Descriptive statistics** refers to methods of summarising data. This topic introduces the notation we’ll use for the sample and population mean and variance of numeric univariate data. We will also introduce quantiles and skewed distributions.

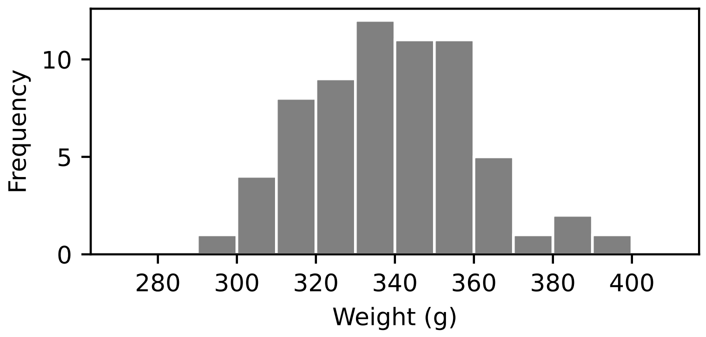
3.2 Distributions of numeric variables

We will focus here on numeric **univariate data**, i.e. data comprising one numeric variable, for example the weights of a number of squirrels that we have captured, weighed, and released. A quick way to start to understand the data is to visualise how it is distributed using a histogram (Figure 3.1a).

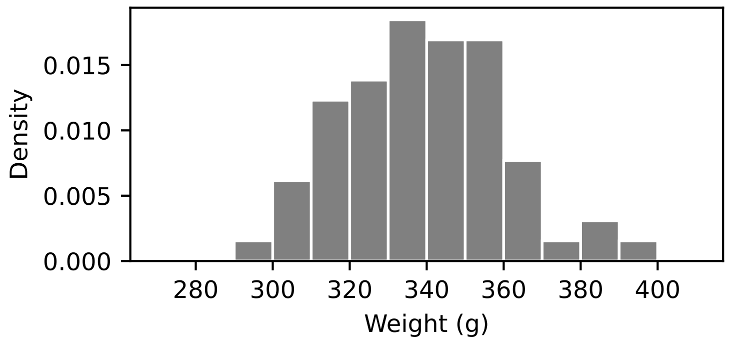
Each bar in a histogram covers an interval of the variable known as a bin. The area of the bar corresponds to how many items are found within the bin. For example in Figure 3.1 the bins are 10g wide. This histogram displays the frequency on the y-axis. From Figure 3.1a we can see that there was 1 squirrel between 290g and 300g in weight, 4 squirrels between 300g and 310g, and so on.

We can also scale the histogram so that the total area under the histogram is equal to 1 (Figure 3.1b). The height of each bar corresponds to the empirical probability density of finding an item in that range. The histogram can be called the **empirical distribution** of the data.

If the histogram has one clear peak we say it is **unimodal**. If it has two peaks it is **bimodal**. Histograms with multiple peaks are **multimodal**. Figure 3.1 appears to be bimodal, but the peak between 380g and 390g is small, and there are not many individuals in this sample, so we should be careful to assume that this distribution is truly representative of all squirrels.



(a) Histogram showing frequency of observations in bins.



(b) Histogram showing probability density of observations in bins.

Figure 3.1: Histograms of weight in grams of a sample of squirrels recorded in the winters of 1985 and 1986 in coniferous woods in North Belgium (Wauters and Dhondt, 1989). The bin width in both histograms is 10g.

3.3 Sample and population mean

Populations and samples It's important to distinguish between **populations** and **samples**. The population is the set of all the things we are interested in, for example, all 400 Scottish wildcats in the Highlands (Figure 3.2). The sample is a subset of the population that we observe – for example 10 wildcats that we trap, measure and release again into the wild.

We refer to the size of the population with N and the size of the sample with n. Note that we don't always know N exactly. In the case of the wildcats, $N = 400$ is an estimate – there is no practical way of counting all the wildcats in Scotland. In other cases we do know N exactly, for example if the population were a pile of exam papers.

Definition of sample mean For a numeric variable x with n observations or instances sampled from a population, $x_1, x_2 \dots x_n$, the **sample mean** is defined as:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.1}$$

Sometimes the sample mean is written informally as $1/n \sum x_i$. When reporting the mean of a set of numbers, one convention is to report to one more decimal place than the accuracy of the x_i's. For example, if the age of 6 cats is $3, 4, 5, 6, 6$ and 7 years, the mean would be reported as 5.2 years, not 5.1666 years. Note that the units of the mean should be quoted; i.e. "5.2 years" *not* just "5.2".

The sample mean is a measure of where the centre of the set of instances is. It is guaranteed to lie between the minimum and maximum x_i. It has the same units as the x_i.

Figure 3.2: Scottish wildcats, *Felis silvestris silvestris*, a critically endangered species. Credit: Peter Trimming / CC BY 2.0)

Population mean For a numeric variable x with N instances, $x_1, x_2 \dots x_N$, the **population mean** is defined as:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.2}$$

With bivariate data (two variables) or multivariate data (more than one variable), we can distinguish between the population means of variables x, y, z by using subscripts: e.g. μ_x, μ_y, μ_z.

The population mean is a measure of where the centre of all instances in the population is. It is guaranteed to lie between the minimum and maximum x_i. It has the same units as the the x_i.

The sample mean is an estimate of the population mean. We will consider how good an estimate it is later in the course when we learn about statistical inference. For now it is enough to know that it depends on how the sample is chosen (randomly or by some other method), and the values of n and N.

3.4 Sample and population median

Definition of median The **sample median** $\tilde{x}$ of a variable x is the "middle" value, when the sampled observations x_i are ordered from smallest to largest. To be more precise, if there are n observations, then the sample median is defined:

$$\tilde{x} = \begin{cases} (\frac{n+1}{2})^{\text{th}} \text{ ordered value, if } n \text{ is odd} \\ \text{mean of } (\frac{n}{2})^{\text{th}} \text{ and } (\frac{n}{2}+1)^{\text{th}} \text{ ordered values, if } n \text{ is even} \end{cases} \tag{3.3}$$

For example, the median age of 6 cats aged $3, 4, 5, 6, 6$ and 7 is 5.5 years. The median age of 5 cats aged $3, 4, 5, 6$ and 7 is 5 years. Note that we should quote the units, as for the mean.

By analogy with the population mean, the **population median** $\tilde{\mu}$ of a variable is the median of the entire population.

Median and mean The mean and median of a sample or population are generally not the same. For example, the mean age of 6 cats aged $3, 4, 5, 6, 6$ and 7 years is 5.2 years, but the median is 5.5 years.

- If a distribution is **symmetric**, $\overline{x} = \tilde{x}$
- If a distribution is **positively skewed**, $\overline{x} > \tilde{x}$
- If a distribution is **negatively skewed**, $\overline{x} < \tilde{x}$

Suppose the age of the cats had been $3, 4, 5, 6, 6$ and 18. The mean age would now be 7 years, but the median is unchanged at 5.5. An instance that appears to be far away from most of the other numbers is called an **outlier**. The example shows that the median is less affected by outliers than the mean. For this reason, the median can be seen as a better way of measuring a typical value of a variable.

It is often worth checking outliers, to make sure that they are real data. Depending on how the data has been collected, an outlier might be due to a faulty sensor, or a mistake in data entry or in the logic of an automated programme collecting data. However, outliers may well be real data, and should not just be removed as a matter of course.

3.5 Variance and standard deviation

Definition of sample variance and sample standard deviation For a numeric variable with n observations or instances, $x_1, x_2 \dots x_n$, the **sample variance** is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{3.4}$$

The **sample standard deviation** is defined as:

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.5}$$

The sample variance and standard deviation give a measure of how spread out the data is. It's an average of a measure of distance of each point from the sample mean (we'll come to why we divide by $n-1$ rather than n later).

One measure of distance we could use is the magnitude (absolute value) of the **deviation from the mean** of each observation: $x_1 - \overline{x}, x_2 - \overline{x}, \dots, x_n - \overline{x}$. We cannot just use the deviations, since they add up to 0 (think about it!). The magnitude of each deviation $|x_i - \overline{x}|$ is guaranteed to be positive. However the average of magnitudes is not as nicely behaved mathematically as the average of the square of the deviations, as defined in Equation 3.4.

It's important to quote the units of the standard deviation and variance. The standard deviation has the same units as the quantity in question and the variance has those units squared.

Definition of population variance and population standard deviation By analogy with the population mean, for a numeric variable in a population of N instances, $x_1, x_2 \dots x_N$, the **population variance** is defined as:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{3.6}$$

The **population standard deviation** is defined as:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3.7}$$

Why the divisor $n-1$ in the sample variance? In short, we'd like the sample variance to be a statistically unbiased estimate of the population variance (we define estimator bias formally in the chapter on Estimation). The squared deviations between the sample mean and the observations $(x_i - \overline{x})^2$ will tend to be smaller than the squared deviations between the population mean and the observations $(x_i - \mu)^2$, so if we divided by n we'd underestimate the sample variance. For the truly interested, we prove that if we divided by n we would have a biased estimate later (Estimation bias and variance).

A second way of thinking about this is that we know that the sum of the deviations is 0:

$$\sum_{i=1}^{n}(\overline{x} - x_i) = 0 \tag{3.8}$$

So if we know all but one ($n-1$) of the deviations, we can use the above equation to deduce the deviation we don't know. We say that this means there are $n-1$ **degrees of freedom**, and it turns out that it makes sense to divide by $n-1$.

In practice, when n is large, the difference doesn't matter much. It's worth being aware that various Python packages have different conventions about dividing by $n-1$ or n. `pandas.Series.std` divides by $n-1$ whereas `numpy.std` divides by n. This behaviour can be changed by specifying the `ddof` parameter in either function.

Scaled quantities Sometimes quantities can be scaled, for example if the units change. If a variable $y = cx$, where c is a scaling constant, then the following relationships hold:

$$\begin{aligned}\overline{y} &= c\overline{x} \\ s_y^2 &= c^2 s_x^2 \\ s_y &= c s_x\end{aligned} \tag{3.9}$$

We'll leave that as an exercise for you to prove.

Another way of writing the variance It's sometimes helpful to rearrange the summation over the squared deviations in the sample or population variance:

$$\begin{aligned}\sum (x_i-\overline{x})^2 &= \sum (x_i^2 - x_i\overline{x} - \overline{x}x_i + \overline{x}^2) \quad \text{expanding} \\ &= \sum x_i^2 - \sum x_i\overline{x} - \sum \overline{x}x_i + \sum \overline{x}^2 \quad \text{splitting up the summation} \\ &= \sum x_i^2 - n\overline{x}\,\overline{x} - n\overline{x}\,\overline{x} + n\overline{x}^2 \\ &= \sum x_i^2 - n\overline{x}^2\end{aligned} \tag{3.10}$$

3.6 Quantiles

Definition of a percentile The yth percentile of a set of numeric observations $x_1, \dots, x_N$ is the value of x_i that is above y% of the values. For example, a baby that is on the 95th percentile for weight will weigh more than 95% of other babies.

The median as the 50th percentile By definition 50% of observations are less than the median, so we could also think of the median as the 50th percentile.

Lower and upper quartiles The 25th percentile is called the **lower quartile** (since it encloses the lower quarter of the distribution) and the 75th percentile is called the **upper quartile**. The difference between the upper and lower quartiles is called the **interquartile range**. The interquartile range is a measure of the spread of the distribution of values.

Quantiles Percentiles and quartiles are all examples of the general concept of **quantiles**. q-Quantiles are the values that divide the population or sample into q separate nearly equally sized groups. Percentiles are 100-quantiles and quartiles are 4-quantiles. Other common uses are deciles (10-quantiles) and quintiles (5-quantiles).

3.7 The mode

Definition of the mode The **mode** is the most frequent element in a set of data. In contrast to all the summary statistics described so far, the mode can be computed for categorical data as well as numeric data. For example the most popular name given to baby boys in 1964 was David – David was therefore the mode of baby boys' names in 1964.

There can be two or more modes in a data set, when there are two or more elements with the largest frequency. Note that a bimodal distribution does generally have two modes, since one peak may be shorter than the other (see for example Figure 3.1a).

Chapter 4

Introduction to data ethics

! Essential reading

Parts 1 and 2 of *An Introduction to Data Ethics* by Shannon Vallor

4.1 Introduction to ethics

Data ethics Even if we're not aware of them, ethical issues arise in our daily interactions with each other and with technology. A way into thinking about data ethics is the potential benefits and harms of data science practices in scenarios.

Example: AI-assisted tax collection In 2022, the French government made several billion euros in late tax returns from people who had swimming pools and weren't reporting them and weren't paying tax on those pools. The Government applied a very basic AI algorithm to Google Maps images to find the pools much faster than could be done by humans checking the images. The tax collected was redistributed though society. But were there any potential downsides of such application of data and algorithms? For example, such algorithms could be used by governments searching for particular individuals who might be participating in demonstrations.

By agreeing to the use of service agreement for this technology, we opt in (often unknowingly) to allowing institutions to take control of our personal data. Although this is legal, it's a big problem because we're compromising our privacy here.

Another example of the social pitfalls of using AI algorithms involves false positives. A well known case involves a man who took a picture of his sick son to send to the doctor and was labelled as a potential child pornographer, and had his Google account blocked. The ethical ramifications of algorithms are a significant issue that is affecting all of us. We will focus on these issues from different aspects.

What is ethics? Ethical principles have been around since the age of Greek philosophers. They provide us with a kind of checklist of what is right and what is wrong in society. Our job as data scientists is to understand and apply that checklist to some of the stages in the data science lifecycle.

Universal and context-specific ethical guidelines Some ethical guidelines are so universal that they're naturally understood, for example, "Thou shalt not kill". That's a rule of behaviour that has been around for thousands of years, and that is a guide to good behaviour that people in society should be able to follow. Other types of ethical behaviour are contextual and depend on the culture, the person and the way in which they are distributed and applied and because ethical reasoning is so subjective. For example, in some countries, no-one stands in a queue. But in the UK that would be considered inappropriate behaviour. So culture and context matter. When building a data science program, it's very difficult to understand the context.

The ambiguity of moral decisions – the trolley problem People are often ambiguous about what are good moral decisions, depending on the context. The trolley problem is one of the tools that philosophers use to talk about the moral dilemmas that come about with ethical reasoning. A runaway train is hurtling towards 5 people on a railway track. You look around, but there's no way of warning them. You notice that you're standing next to a lever that you can operate. You can divert the train onto another track and save the people. But there is a problem: there is also a person on the other track. If you hit the lever and divert the train, then they die. If you do nothing then 5 people die, but if you pull the lever then 5 people are spared but 1 person dies. About 50% of the people in different places all over the world were asked how they would behave in this hypothetical scenario are willing to switch the points, for the net benefit of saving four people.

Philosophers consider variations of this problem to understand how people think about different contexts. Now imagine there are no points any more, but there's a man standing on top of a bridge. Now the only way to save these five people is by pushing the man on the tracks, thereby stopping the train with his body. The number of people willing to push the man off the bridge is fewer than those willing to switch the points, even though the net effect is the same (saving 5 people at the expense of one person).

Relevance of the trolley problem This is much research on understanding how should autonomous vehicles should behave in situations when they might kill people. Suppose the brakes have failed in an autonomous vehicle that is approaching a junction that a school bus is crossing. In order to avoid the school bus, the vehicle needs to slam into a tree and kill the driver. If you think about utility, that is the rational outcome[1].

When surveyed, most people agreed the autonomous vehicles should take moral decisions for the benefit of society and the common good. But people in the survey weren't willing to buy such a car. The trolley problem has a real, strong relevance to how computers should be making decisions. This example demonstrates that data ethics is a wide, surprisingly complex area of research that requires philosophers and computer scientists and machine learning people to work together.

Ethical reasoning is relevant to all the stages of the data science life cycle, from when data is collected, how data is used, how data is distributed, and even how it is controlled. We shall touch on several of these aspects.

4.2 Data protection and privacy

Privacy The UK is networked with more cameras on the streets than any other Western country, and those cameras are used by the police and other enforcing law agents to help keep the peace. We as the public are willing to give up some of our privacy in return for order and discipline in our lives.

But how far are we willing to go to maintain order? Would we also allow cameras into our home, for example, if we knew that they could be used by police to help track potential criminals? Many of us may not allow the use of surveillance cameras in our home. Indeed, concerns over privacy, like many social issues, are culture dependent. This is why there is no "one size fits all" approach to privacy, and the request of social media sites to provide them with default permission to access our data is too strict.

GDPR The EU has a very successful and has a very successful list of regulations on data protection and privacy called the General Data Protection Regulation Rights or GDPR `https://gdpr-info.eu/`. GDPR lists regulations that concern rights of the data subjects, duties of data controllers or processors, transfers of personal data to third countries and more. Importantly, it also details liability or penalties for breach of rights. GDPR has also been widely applied outside the EU and is considered to be a gold standard in data protection.

In particular, GDPR allows individuals the right to access and rectify data that concerns them, and even the right to erase data This is also known as the 'right to be forgotten'. This right can be exercised if personal data is no longer necessary for its original purpose, or that it harms someone individual interests and there is no overriding legitimate interest to keep it. For example, Germany's highest court has ruled that A German man convicted of murder in 1982 has the right to have his name removed from online search results,

Interventions As data scientists, we should be aware that data is used by algorithms I'm intervening in your lives and making you engage in behaviour that you would not have done otherwise. For example, convenience

[1]You can participate in an experiment presenting similar scenarios at `https://www.moralmachine.net/`.

chain stores in the USA have been known to track consumer behaviour for the purpose of recommending products for them. Ride sharing apps have been known to use intervention methods to get drivers to stay longer on the road, when they are predicted to quit and go home. Video sharing apps use recommendation algorithms to lure people to consuming more and more content, while this content becomes more radical or extreme in nature. This phenomenon has been coined "down the rabbit hole" `https://www.anewseducation.com/post/youtube-rabbit-holes`.

personal data belonging to millions of Facebook users was collected without their consent by British consulting firm Cambridge Analytica, predominantly to be used for political advertising.

We will also discuss possible ways of mitigating the influence of AI algorithms on our behaviour. There are several rules of behaviour that we can apply to respect people's privacy and autonomy, such as

- being transparent about whether AI is being used to recommend or to motivate users.
- Allow people to opt out at any stage from receiving AI generated interventions.

4.3 Bias

The definition of bias is an inclination or prejudice for or against one person or group, especially in a way considered to be unfair. It turns out that bias can be exhibited not only by people, but also by computers. Algorithmic bias refers to attributes of an algorithm that create unfair outcomes. When it does this, it unfairly favours someone or something over another person or thing.

Algorithmic Bias Algorithmic bias describes systematic and repeatable errors in a computer system that create "unfair" outcomes, such as "privileging" one category over another in ways different from the intended function of the algorithm.

Bias can emerge from many factors, including but not limited to the design of the algorithm or the unintended or unanticipated use or decisions relating to the way data is coded, collected, selected or used to train the algorithm.

Machine learning algorithms can introduce bias at different stages in the data science life cycle; they may provide biased recommendations and decisions. Noteworthy examples of machine bias that have been widely reported in the media include:

- The COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) algorithm used in US court systems to predict the likelihood that a defendant would become a re-offender. This model predicted twice as many false positives for black offenders than white offenders.
- In 2015, Amazon realized that their algorithm used for hiring employees was found to be biased against women.
- In 2019, Facebook was found to be in violation of the U.S. constitution, by allowing its advertisers to deliberately target adverts according to gender, race, and religion, all of which are protected classes under the country's legal system.

How can we mitigate machine bias? Unfortunately, there are no quick answers to this question. Humans are the ultimate source of bias, as they are the ones generating the data and writing the algorithms that use the data to learn and make decisions. There is a lot of work (outside the scope of this course) on detecting fairness and bias in machine learning. It is also clear that regulatory practices should be developed that limit algorithm discrimination and also provide guarantees such as the right to explanation. We will discuss several of these issues in the data ethics workshop.

4.4 Case Study: The effect of gender on credit scoring

Case study. The Law of Equal Opportunities or Unintended Consequences: the impact of unisex risk assessment in consumer credit

We will present a case study by Dr. Galina Andreeva who is from the business school here at the University of Edinburgh. This paper is about credit scoring, which is how banks assess the risk of individuals applying for loans. This is a procedure that's been around for a long time. Banks compute a score for each application based on different attributes, and then they rank all the applications and choose a cut-off. Those with a score above that cut-off will receive the loan and those with a score below that cut-off will not receive the loan. Nowadays, banks use computer algorithms to compute credit scores using attributes like age, time in current address, occupation, and so forth. The natural question to ask is what attributes (features) we should allow these algorithms to use in order to avoid bias. Legislation prohibits characteristics from being used in credit scoring (race, colour, national origin, gender, marital status, religion, receipt of public assistance, or exercise of consumer protection rights). This is called the law of **equal opportunity**. Protected attributes include disability, gender reassignment, marriage and civil partnership, pregnancy and maternity, race, religion or belief, sex, sexual orientation. Age has special status and can be used in some cases (credit scoring is one).

The intent of the legislator is that by practising equal opportunity in credit scoring, we will achieve **equal outcome**, which simply means that two individuals who only differ on protected attributes should not differ in the likelihood to obtain a loan.

The study showed this was not the case in practice. Specifically, equal treatment was not fulfilled, in that the model learned to distinguish applications based on attributes that are proxies for gender, such as salary (unfortunately, women still earn less than men on average).

The study also showed that gender itself is a statistically significant predictor of credit risk. If we allowed the model to train on gender, we could get a more accurate prediction of credit risk. This means that women are "paying" for the fact that their gender is a protected attribute, in that their cut-off is higher than it would be if the model was allowed to distinguish on gender (and conversely for men). The conclusion is that existing law is not effective in promoting equality when it comes to algorithms.

Related Workshop: Data ethics discussion

Chapter 5

Exploratory data analysis and communication of results

! Essential reading

The Big Book of Dashboards (Wexler et al, 2017), Chapter 1

See the Resource List for a link to this resource.

5.1 Visualisation for exploratory data analysis

Exploratory data analysis Once we have got a dataset, the next step in a data science project is to explore the data. The objectives of **exploratory data analysis** (EDA) are to:

- Identify any possible anomalies in the data, e.g. missing data or data points that appear unlikely
- Suggest hypotheses about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys or experiments

In exploratory data analysis, we should try to avoid any preconceived ideas about relationships in the data, but we should use our pre-existing knowledge to assess whether data make sense. For example, if we come across a temperature of over 100 °C in a time series of temperature recorded outside in Edinburgh, we should be suspicious that there has been an error in the sensor, or the processing of the data we have received.

Why is visualisation important? We define **visualisation** as the process of conveying information through graphical representations of data. Visualisation is a crucial part of our own exploratory data analysis, and is also vitally important for communicating results to others.

The aims of communicating data are to

- Present data and ideas
- Explain and inform
- Provide evidence and support
- Influence and persuade

This may sound obvious, but it is surprising how a changing the presentation of a dataset can make features of the dataset become more apparent.

The influential statistician John Tukey explained the power of visualisation thus:

> The simple graph has brought more information to the data analyst's mind than any other device. The greatest value of a picture is when it forces us to notice what we never expected to see.

It has the property of forcing "us to notice what we never expected to see" that makes visualisation so powerful. For example, suppose we looked at the series of ambient temperature readings mentioned above as numbers. We would be unlikely to spot the reading of 100 °C. However, if we plotted the series with time on the x-axis and temperature on the y-axis, this data point would stand out at us.

Why does visualisation work? Put simply, the human visual system is very good at identifying what are called **preattentive attributes** in scenes. Wexler et al. (2017) list 9 preattentive features:

- Length
- Width
- Position
- Size
- Shape
- Curvature
- Added marks
- Enclosure
- Colour value
- Colour hue
- Spatial grouping

We use these features to construct visualisations to show the data in a which aspects of the data "pop-out" to us. It is also possible to create visualisations that mislead or confuse. Creating visualisations that truthfully and informatively represent the data is an art that takes practice to acquire.

How we create visualisations Before the computer age, visualisation was a painstaking process of drawing lines accurately on paper. The earliest work that we would recognise as modern visualisations were in the late 18th century by the Scottish economist William Playfair (not the architect), who created visualisations to illustrate various economic data. Nowadays visualisations can be created by computer packages such as Excel, or via cloud-based packages such as Power BI and Tableau.

In this course we will focus (in the labs) on creating visualisations programatically using Python's Matplotlib and Seaborn packages (though we could also have used R). Creating plots using computer code can seem more time-consuming than doing so using a graphical package. However, an advantage is that we can reproduce the steps taken to analyse data and produce a visualisation. We also have fine control over the design of our visualisations.

5.2 Principles of visualisation

Various authors have proposed sets of principles of visualisation to guide the creation or assessment of visualisations. We have constructed a simplified set of principles based on those of Tufte (1982) and Schwabish (2021).

Principle 1: Show the data

The aim is to show as much of the data as possible without leading to a confusing visualisation. There are often multiple ways of representing the same dataset, and no "right" answer. The following guidance on arranging the graphical elements of the plot should help you to show as much of the data as possible:

- **Choose an appropriate plot type.** Some basic types are:
  - Bar charts: good for plotting numeric variables associated with categorical or ordinal variables, for example the mean weight (numeric variable) of male and female (categorical variable) squirrels.
  - Line charts: for showing trends of numerical variables over time (a numerical variable).
  - Scatterplots: show the relationship between two numeric variables.
  - Histograms: are good for showing the distribution of a single numeric variable, for example the weights of male squirrels. The *area* of each bar in a **histogram** shows the **frequency** (i.e. number) of observations within a set of **bins** (i.e. intervals). The size of bins is your choice: up to a point smaller bin sizes show more detail, but a very small bin size will obscure the distribution. Bins can be of different sizes – because we're plotting the area, this won't over-represent counts in large bins. It's also possible to normalise histograms by making the area of each bar equal to the **density**, i.e. the frequency in each bin by the total number of observations. In this case the total area in the histogram is equal to 1.
  - Density plots: also show the distribution of a numeric variable and can be seen as a smoothed histogram. The density is an estimate of the probability density function underlying the distribution, and can be generated using kernel density estimation: the estimated density at each point is computed by placing normal distributions of a particular width (called the bandwidth) around each point. Analogous to the bin width in histograms, narrower bandwidths lead to a more bumpy appearance, and wider ones a smoother appearance. Packages such as Seaborn set the bandwidth automatically, but it's possible to adjust it.
  - Boxplots: represent the distribution of a numeric variable for multiple categories, e.g. the weights of male and female squirrels. Boxplots (also known as box and whisker plots) are one-dimensional representations of a distribution in which the box extends from the lower (first) to the upper (fourth) quartile values of the data, while the line across the box represents the median. The 'whiskers', the lines extending from the box, can represent different things, as described in the Wikipedia article on boxplots. By default, Matplotlib defines the end of the upper whisker as the value of the largest data point that lies within 1.5 times the interquartile range from the upper quartile, and the lower whisker as the value of the smallest data point that lies within 1.5 times the interquartile range from the lower quartile. Data points that lie outwith the whiskers are called outliers, and are represented by dots or circles. Since the whiskers can represent multiple statistics, ideally their meaning should be indicated in the plot caption.
- **Show multiple variables by using length, shape, size and colour:**
  - The above plots are univariate or bivariate, since they display one or two variables. We can use shape and colour to create extra dimensions for categorical variables. For example, in a scatterplot of squirrel weight versus length, we can indicate sex using colour, thus displaying 3 variables. We could also indicate age categories by changing the size or the shape of the markers (4 variables). However, we must be careful that adding information using marker properties does not detract from the plot.
  - Barcharts can be extended to two categorical variables and one numerical variable by using colour.
- **Use colour effectively.** (Wexler et al., 2017, pp. 14–18)
  - Choose an appropriate colour scale, depending on if the data is sequential, diverging or categorical.
  - Colour can also be used to highlight features in the plot, e.g. the largest two bars in a bar plot.
- **Encourage the eye to compare several pieces of data**, e.g. by using multiple plots with the same scale.

  - Wexler et al. (2017), p. 31, is a nice example of how this can work better than using multiple symbols on a plot (p. 30).

- **Present many numbers in a small space**
  - A boxplot takes up as much space as a barplot, but conveys more information. For example, a boxplot of the squirrel's weight versus sex shows information about the distribution of the weight as well as the median weight.

- **Choose appropriate transforms**
  - Sometimes it can make sense to **transform** data so that features of it are clearer. For example, plotting the value of Bitcoin over time shows very little detail about the early history of the currency, when it was not valuable. However, plotting the log of the value of Bitcoin on the y-axis allows this detail to be seen.

Principle 2: Make the meaning of the data clear

A visualisation is meaningless if it's not labelled. Every plot should have:

- Title or caption
- Axis labels as English words
- Units given, where appropriate (e.g. "Length (mm)" *not just* "Length")
- All variables labelled – e.g. a legend indicating the colours used to represent squirrel sex
- Use graphical and textual annotation – e.g. it can be helpful to highlight a time series with events that you know about

Principle 3: Avoid distorting what the data have to say

Choices in visualisation design can lead to the instant impression given by preattentive processing of the visualisation being quite different to the numbers in the dataset. Tufte (1982) measures the level of distortion in a visualisation by the "Lie factor":

$$\text{Lie factor} = \frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$$

Here by "size of effect" we mean the relative change from the "control" condition. For example, if there is an increase of 50% over a control condition, we say the **effect size** is 50%. A lie factor of 1 means no distortion – the lie factor indicates how much the visualisation has exaggerated the effect size in the data.

Example of lie factor calculation

Consider the left-hand panel of Figure 1 in Kramer et al. (2014), which shows the percentage of words written by Facebook users in a week that were classified as emotionally positive for users in a control condition, or in an experimental condition in which 10% of users' friends' posts containing negative words were omitted from their feeds. Suppose that the quantity measured in the control condition is y_C and the quantity measured in the experimental condition is y_E. Then

$$\text{size of effect in data} = \frac{y_\mathrm{E} - y_\mathrm{C}}{y_\mathrm{C}}$$

In the "Negativity reduced" condition (left), $y_\mathrm{C} = 5.24\%$ words, and $y_\mathrm{E} = 5.30\%$ words. Therefore, the effect size in the data is (5.30-5.24)/5.24 = 1.132%. However, if we look at the size of the bars in the data, the control bar 0.24 high, and the experimental bar is 0.30 high. Thus, the effect size in the graphic is (0.30-0.24)/0.24 = 25%. Therefore, Tufte's lie factor is 25/1.132 = 21.8.

The following guidelines should help to avoid distorting the data:

- **Use appropriate scales and baselines**
  - A very common problem is that the baseline (i.e. the lowest point on the $y-$axis) in a barchart is not zero. This can lead to small differences appearing large, and will cause the lie factor to be different from 1. However, in some cases a non-zero baseline is justified, for example in time series when it is the absolute changes over time rather than proportional differences that are important to see.
- **Be aware of limitations of our perception of size**
  - Although marker area can be useful for indicating categories, humans are not very good at relating the area to a quantity – we are much better at comparing lengths.

Baselines and the lie factor

Suppose the baseline used in the graph is y_0. Then

$$\text{size of effect in graphic} = \frac{(y_\mathrm{E} - y_0) - (y_\mathrm{C} - y_0)}{y_\mathrm{C} - y_0} = \frac{y_\mathrm{E} - y_\mathrm{C}}{y_\mathrm{C} - y_0}$$

We substitute the two equations into the equation for the lie factor to give:

$$\text{Lie factor} = \frac{y_\mathrm{C}}{y_\mathrm{C} - y_0}$$

If $y_0 = 0$, i.e. when the baseline of the variable we are measuring is at zero", we can see that the lie factor is 1.

For positive y_C with a baseline above 0, the lie factor will be above 1, i.e. the size of the effect will be exaggerated. We can make the lie factor less than one (i.e. understating the size of the effect) by having a negative baseline – but this rarely happens.

Principle 4: Make the data accessible

A visualisation is meaningless if it's illegible and loses impact if it's difficult to read. To ensure data is accessible:

- **Make sure text is legible**, i.e. font size of minimum 8 points in a PDF, or about 20 points in a presentation. (It is surprising how often talks are given in which it's impossible to read the labels on plots even from the front row.)
- **Use colours that work for people with colour-vision deficiency.** Wexler et al. (2017), Chapter 1 has an excellent introduction to using colour in visualisations.

If you shrink or expand a figure, the font size changes

Suppose you are writing a report in which the width of the page is 8 inches, with 1 inch left and right margins. The width of the body text is therefore 6 inches. You generate a figure that has a `figsize` of `(6, 4)`, and a `fontsize` of 8 points, save the figure to file, and include it in your document. Since `figsize=(6, 4)` means "6 inches wide and 4 inches tall", when you include the figure, the font size remains 8 points. All is well.

Suppose now you are having difficulties fitting all the numbers on the plot in. You decide to set `figsize=(12, 8)` (12 inches wide and 8 inches high) and keep the font size at 8 point. Bingo! Your numbers now fit. You put the image in the document so that it fills the width of the page. *But now the actual font size is 4 points, because you have shrunk the image by a factor of two to fit it into the page.* **Your plot will not be accessible to some people. This is a very common mistake.**

Principle 5: Focus on the content

Give the viewer's brain as little work to do as possible.

- No chartjunk – e.g. colours that don't have any meaning.
- Reduce clutter
- Consistent colours between plots in a study
- Correct spelling

5.3 Effective and ineffective visualisations

In his book *The visual display of quantitative information*, originally published in 1982, Tufte (2001) highlighted some outstanding examples of visualisation from the time of Playfair onwards, criticised some trends in visualisation that he identified as being prevalent by 1980, and proposed some principles for graphical integrity and good visualisation. His book is inspirational, but we should also bear in mind that some of the visualisations he most admires would be impossible to create in Matplotlib or Seaborn, since they lack some of the flexibility of hand-drawing or using a graphics package. Also, some of his principles were reacting against problems that occur less often, due to the standardised nature of visualisation software.

- Ineffective visualisations
  - Scale distortions
  - Ineffective colours
  - Graph junk
- Effective visualisations
  1. Graphical integrity - no scale distortions
  2. Simple – Edward Tufte and Data-ink ratio, avoid chart junk
  3. Use the right display
  4. Use colour strategically – Luminence and colour blindness
  5. Tell a story with data

Related Python Lab: Data Representation I – Matplotlib

`https://github.com/Inf2-FDS/FDS-S1-03-data-representation-01`

- Matplotlib basics
- Basic visualisation of numerical and categorical distributions
- Scatter plots and line graphs
- Histograms and bar chars
- Overlay charts
- Functions and tables

Related Python Lab: Data representation II: Distributions

`https://github.com/Inf2-FDS/FDS-S1-05-data-representation-02-distributions`

- Mean, mode, median, outliers, percentiles,
- box plots, variance and standard deviation, missing values,
- Seaborn pairplot
- Maybe some additional exercises combining previous things - try to use examples related to our School's strengths.

Related Workshop: Visualisation

Chapter 6

Data collection and statistical relationships

6.1 Obtaining data hosted online

Methods of obtaining online data There are a number of ways of obtaining data hosted online.

- Downloaded as structured files (e.g. tablular data, or data in JSON format) from websites, e.g. scientific, government or charity data repositories
- Via an **API** or database
- **web scraping** – the process of automating the process of extracting data from a web page.

These are listed in order of difficulty. It's generally better to start with the easier options first – there's no need to scrape data if it's already available in a structured format.

The first rule of web scraping is don't do it unless you have to. Data is increasingly available in structured formats such as CSV and JSON. Web scraping seems cool, but it's also time-consuming and fiddly. It pays to spend some time searching for files in structured formats, in which the data may well be in a cleaner format than on a web page. However, if you can't find the data you need in a structured format, but it is available on a web page, then web scraping can be helpful.

Ethical and legal considerations in online collection It is worth remembering that using online data involves a relationship between you and the data creator, and possibly the people who funded the data collection, so we need to think a little about law and ethics.

Licences for downloaded file If you're downloading data files, it's likely they will have been issued with a data license, which governs what you are allowed to do with the data, and how you can publish any work that you produce using that data. A few common types of license are:

Creative Commons These general-purpose licenses can come with conditions, such as that the creator must be credited (BY) or that only non-commercial uses are allowed (NC). The CC0 license is very permissive, meaning that the owner has no control of the data, not even requiring data users to attribute the data creator – but it is always good practice to attribute sources.

Open Data Commons Similar to the Creative Commons licences, but designed for data.

Open Government Licence A licence created by public bodies in the UK.

You should check the license or copyright statement, if there is one. This section of *The Turing Way* (Turing Way Community, 2022) has more information about data licences.

API conditions Sites with an API (e.g. Twitter) have conditions under which you can take the data – you should respect those conditions if they are not already enforced by the API.

Figure 6.1: Red squirrel, *Sciurus vulgaris*. Credit: Peter Trimming / CC BY 2.0)

Ethical web-scraping As with data files, the owner of the website you wish to scrape may have invested considerable time and money in creating their site. It's possible that by web scraping you could have an adverse effect on either their intellectual property or their web server.

The law around web scraping is not always clear (Davies, 2020) but you should always check the terms and conditions of the website before starting scraping. For example, the Copyright Policy of the *Financial Times* website says you cannot "Frame, harvest or scrape FT content or otherwise access FT content for similar purposes." In contrast, the *Time Out* website terms and conditions are more permissive.

Beyond the legal restrictions for a specific website, we should also consider general ethical principles of web scraping. Densmore (2017) suggests a number of rules for those undertaking web scraping, including:

- using an API if available
- requesting data at a reasonable rate (not hoarding bandwidth)
- respect content rights, and not passing date off as one's own
- giving back to the data owner when possible, including attributing the data owner in any publication
- scrape only to create value from data, not to duplicate it

6.2 Correlation

In the topic on Descriptive statistics, we considered summary statistics of one variable, for example the weight of a wildcat, or another wild animal, such as the red squirrel (Figure 6.1). In data science and statistics we're often interested in statistical relationships between two or more variables. In this section we'll discuss the most basic statistical relationships: covariance and correlation.

Covariance Suppose that as well as measuring the weight in grams x_i of the ith squirrel, we also measure its length in millimetres y_i (Figure 6.2). The **sample covariance** is defined:

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{6.1}$$

We expect that squirrels that are longer than average will also be heavier than average, i.e. both $x_i - \overline{x}$ and $y_i - \overline{y}$ are positive. Thus, for these squirrels $(x_i - \overline{x})(y_i - \overline{y})$ will be positive. We also expect that squirrels that

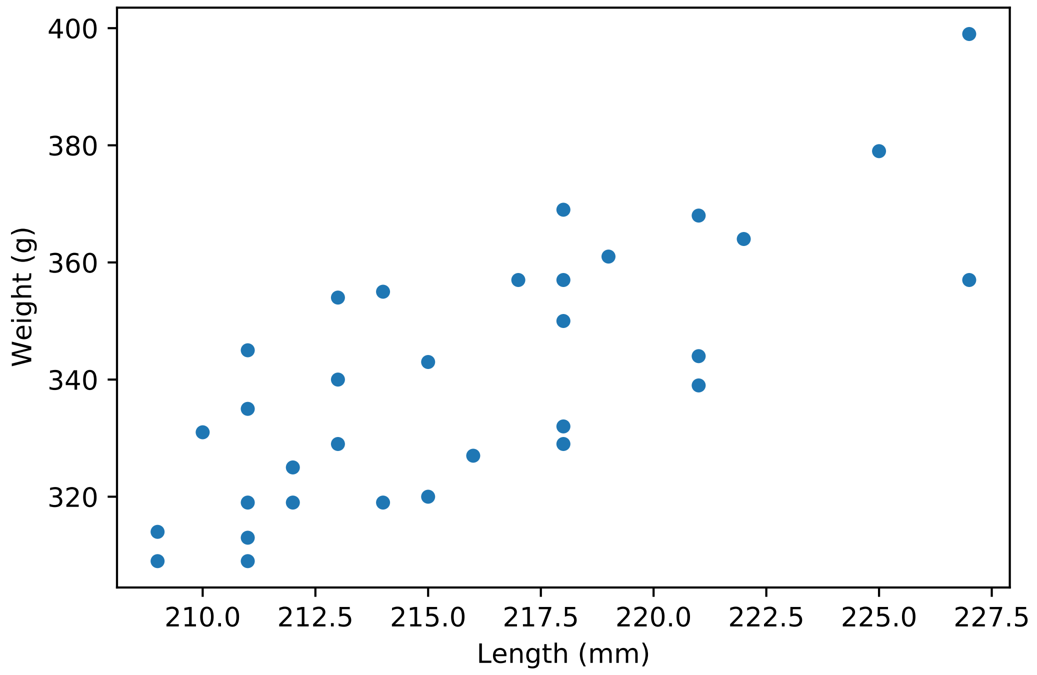


Figure 6.2: Weight in grams versus length in millimetres of a sample of female squirrels recorded in the winters of 1985 and 1986 in coniferous woods in North Belgium (Wauters and Dhondt, 1989). The correlation coefficient of the length and weight is $r = 0.77$.

are shorter than average will be lighter than average. For these squirrels both contributions $x_i - \overline{x}$ and $y_i - \overline{y}$ will be negative, but the product $(x_i - \overline{x})(y_i - \overline{y})$ will be positive. We should thus expect that the covariance of squirrel length and weight is positive.

If one quantity gets smaller as the other one gets bigger, then, by similar reasoning, the covariance is negative.

There are a few points to note about the covariance:

1. Its units are the product of the units of the two variables; in this example the units would be g · mm.
2. It depends on the scaling of the variables; suppose we measured the squirrels' weight in kilograms (instead of grams) and their length in centimetres (instead of millimetres). The covariance would be 10,000 times smaller, even though the relationship between the variables remains the same.

We don't want a measure of how strongly quantities are related to depend on the units they are measured in, so the second point is a problem.

Correlation coefficient The **sample correlation coefficient** (also known as Pearson's correlation coefficient[1]) addresses this problem by dividing the covariance by the product of the standard deviations of the two quantities:

$$r = \frac{s_{xy}}{s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{6.2}$$

It has a number of properties:

[1]Karl Pearson (1857–1936) was a remarkable polymath – and a eugenicist. Although he invented many statistical concepts, the formula of the correlation coefficient was originally published before he was born.

1. It ranges between -1 and 1.
2. $r = 1$ corresponds to all points lying on straight line sloping upwards and $r = -1$ corresponds to all points lying on a straight line sloping downwards.
3. It has no units, i.e. it is dimensionless
4. It is independent of the units in which x and y are measured
5. r remains the same if we swap the labels of x and y

To convince yourself of the first property, imagine what would happen if all $y_i = cx_i$.

Standardised variables Another way of looking at the correlation coefficient is in terms of **standardised variables**. The standardised version of a variable x is, by convention, denoted by z, and is also refered to as a z-score. The standardised version of ith instance is defined as:

$$z_i = \frac{x_i - \overline{x}}{s_x} \tag{6.3}$$

The standardised variable z_i has several nice properties:

1. It has zero mean, i.e. $\overline{z} = 0$
2. It has unit variance, i.e. $s_z^2 = 1$
3. It is dimensionless (has no units)
4. The covariance of two standardised variables is equal to the correlation coefficient.

We'll leave you to prove the first two points. To see the third point, suppose that z is the standardised version of x and u is the standardised version of y, which we substitute into Equation 6.1 for covariance:

$$\begin{aligned}
s_{uz} &= \frac{1}{n}\sum_{i=1}^{n} \frac{z_i - \overline{z}}{s_z}\frac{u_i - \overline{u}}{s_u} \\
&= \frac{1}{n}\sum_{i=1}^{n} z_i u_i \quad \text{by properties 1 and 2} \\
&= \frac{1}{n}\sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{s_x}\right)\left(\frac{y_i - \overline{y}}{s_y}\right) \\
&= \frac{1}{s_x s_y}\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \\
&= \frac{s_{xy}}{s_x s_y} \quad \text{which is the definition of the correlation coefficient}
\end{aligned} \tag{6.4}$$

What's hiding in a correlation coefficient? With multivariate numeric data, a very helpful technique can be to look at the correlation coefficient of every pair of variables. These can be plotted as a heatmap, quickly showing where there might be interesting relationships (i.e. correlation coefficients close to 1 or -1). We'll use this technique in the labs.

But before using correlation heatmaps, we should take a look at Figure 6.3. Can you guess roughly what the correlation coefficient is in each plot? Before going to the next the page, try to think about how you would describe the data in each plot in words.

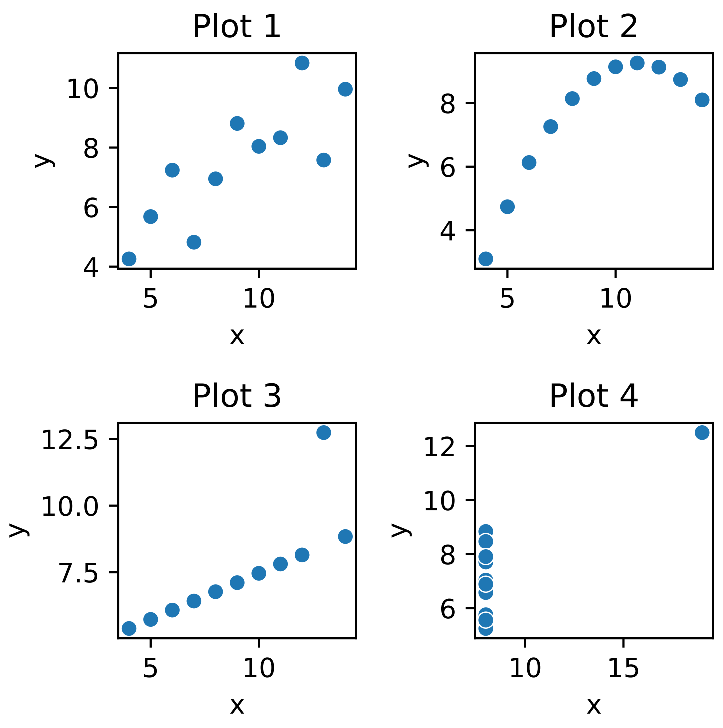


Figure 6.3: What are the correlation coefficients in these plots? "Anscombe's quartet" (Anscombe, 1973)

Well, the answer is they all have the same correlation coefficient, $r = 0.82$. But the visualisation indicates that something quite different is happening in each one:

1. Basically a linear relation, with some noise around it – quite like the squirrel length and weight
2. It looks like there is a very precise nonlinear (perhaps quadratic) relationship between the variables here
3. It looks like there would be a linear relationship with $r = 1$ if we took out the outlier point
4. Again there is something suspicious about the outlier point

6.3 Limitations of statistical relationships

Correlation and causation You've probably heard the phrase "correlation is not causation". Let's define what we mean by correlation and causation:

Two variables are **correlated** when knowing the value of one gives you information about the value of the other variable.

Two variables are **causally related** when changing the value of one of the variables affects the value of the other variable.

Spurious correlations Suppose we have a correlation between two variables, and moreover, we're happy that it looks linear – does this indicate that one variable is causing a change in the other variable? For example the number of Nicolas Cage and number of people drowned in swimming pools. **Spurious correlations** are correlations that arise by chance between two unrelated variables.

Biases in data **Selection or sampling bias** is when the sample chosen is not representative of the population under investigation.

Confounding variables A **confounding variable** is a variable that influences two or more variables, meaning that the two variables are correlated, even if there may be no causal relationship between them.

Establishing causation

- Observational data
  - Inference from observational data
  - Correlation vs causation
  - Case study: miasmas and cholera, John Snow's "grand experiment"
  - Confounding factors
- Experiments
  - Example: A/B Testing

Related Python Lab: Web-scraping

`https://github.com/Inf2-FDS/weekS2-06-web-scraping`

Part II

Introduction to Machine Learning

Chapter 7

Supervised learning: Classification with Nearest neighbours

Further reading (not examinable)

Hastie et al. (2009) pp 463–471

7.1 Classification

The classification problem Suppose a bank has data on previous customers it has given loans to, including variables such as their income, housing status and employment status. Each of these sets of variables – also referred to as **feature vectors** – has a **label**, indicating whether the customer did or didn't pay back their loan. The bank might want to predict whether a new customer will be able pay back a bank loan from their features, i.e. to predict whether they belong to the class of customers who paid or the class of customers who didn't pay. This is an example of the **classification problem**, which we define as the problem of predicting the correct category label ("class") for an unlabelled input feature vector.

Supervised and unsupervised learning Classification is an example of a supervised learning process. In a **supervised learning** process, there is a **training set** of data in which each data item has a number of **features** and a known **label**. The goal of supervised learning is to predict the label of an item that has not been previously seen from its features. In contrast, in **unsupervised learning** processes, the training set does not contain any labels, and the goal is to learn something about the structure of the data. We will return to unsupervised learning in a later topic.

Visualising the classification problem To visualise the classification problem, we'll use a toy example: the fruit data set, collected by Iain Murray (Murray, 2006). He bought pieces of fruit and measured their height and width (features) and noted the type of fruit (the label). Figure 7.1 visualises the data. In the context of the fruit, the classification problem is using this dataset to build a machine to predict the class of a piece of unidentified fruit automatically just by measuring its width and height. We will refer to the feature vector of this unidentified fruit as the **test point**.

Definition of a classifier To solve a specific classification problem, we construct a **classifier**. A classifier is a function that takes a feature vector $\mathbf{x}$ and returns a class c where c is a member of a set $\mathcal{C}$. In principle, we can construct a classifier in any way we want to, as long as it matches this definition. Regardless of how the classifier is constructed, it will have two important properties: **decision boundaries** and **classification errors**.

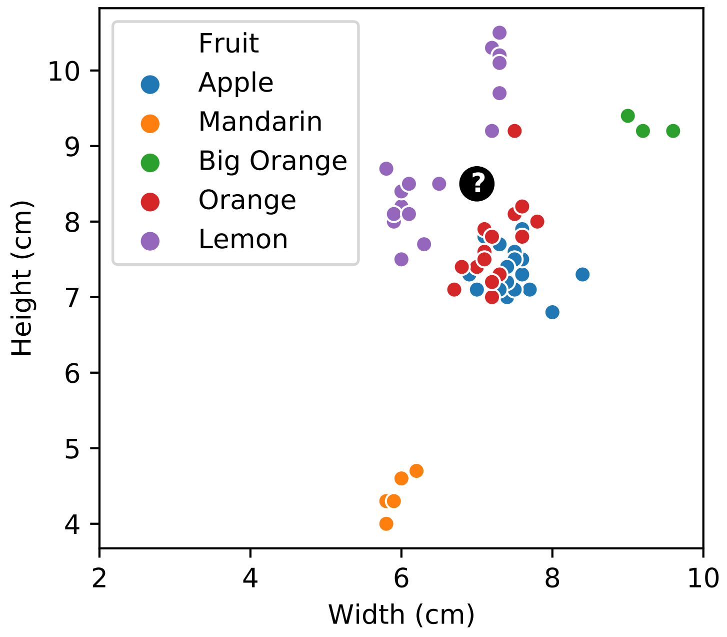


Figure 7.1: The supervised learning problem, as applied to fruit. We are given the labels of the fruit with various widths and heights. We are then presented with an unknown piece of fruit with given width and height (the test point, represented by the question mark). The task is then to predict what type of fruit it is.

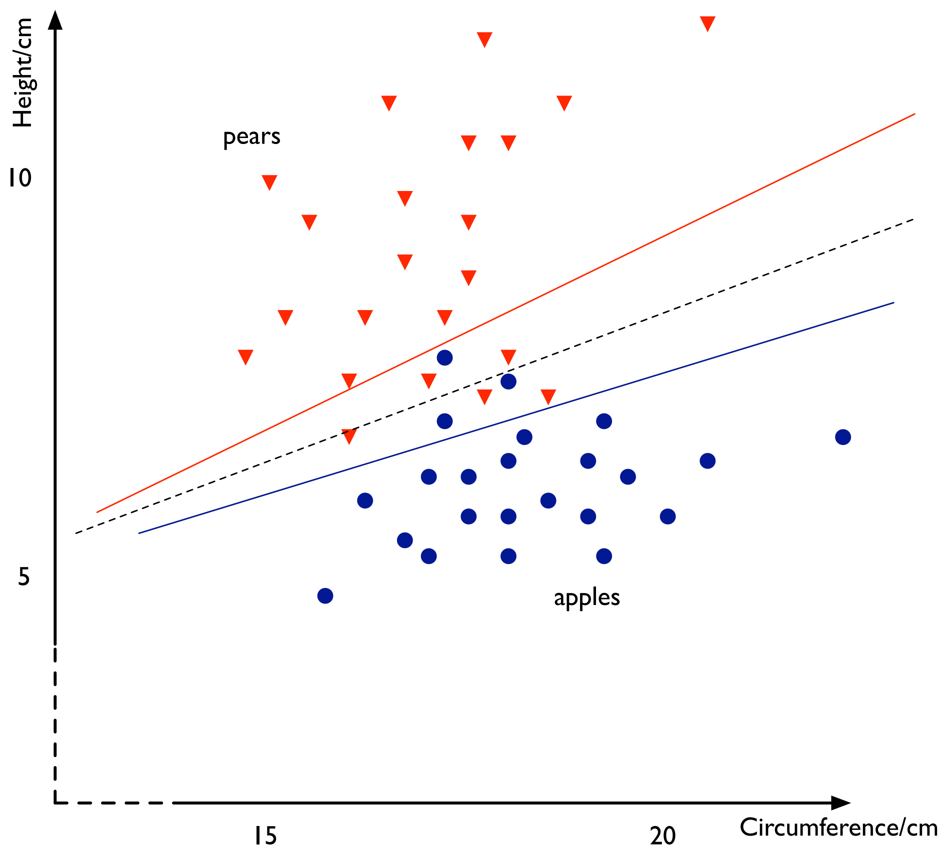


Figure 7.2: Training data for apples and pears. It is not possible to draw a straight line to perfectly separate the classes in this feature space. Three possible lines are drawn, but each results in a number of misclassifications.

Decision boundaries Consider the problem of determining apples from pears. In this example case we have two features for each piece of fruit: its circumference (at the widest point) and its height, each measured in cm. Apples are 'more spherical' than pears which tend to be longer than they are wide. But some pears are relatively short, and some apples are taller than expected. In this case we have an input vector of the form $\mathbf{x} = (x^{(1)}, x^{(2)})^T$, where $x^{(1)}$ is the circumference and $x^{(2)}$ the height. The class c can take two values A or P (standing for apples and pears).

We have a set of training data: height and circumference measurements, along with class labels. In Figure 7.2 we plot this two-dimensional training data for the two classes. We can see that it is not possible to draw a straight line to separate the two classes.

We now have three new, unlabelled examples which we would like to classify (represented as stars in Figure 7.3):

- $(16, 10)$: all the training data in the region of this point is classified as P, so we classify this point as P.
- $(19, 6)$: looking at the training data it seems obvious to class this as A.
- $(18, 7)$: it's not obvious in which class this example should be classified; the feature vector gives us evidence whether we have an apple or a pear, but does not enable us to make an unambiguous classification.

We can draw a straight line such that one side of it corresponds to one class and the other side to the other – as in the three possible lines shown in Figure 7.2. Such a line is called a **decision boundary**; if the data was three-dimensional, then the decision boundary would be defined by a plane. For one-dimensional data, a decision boundary can simply be a point on the real line. Intuitively it seems possible to find an optimal decision boundary, based on minimising the number of misclassifications in the training set.

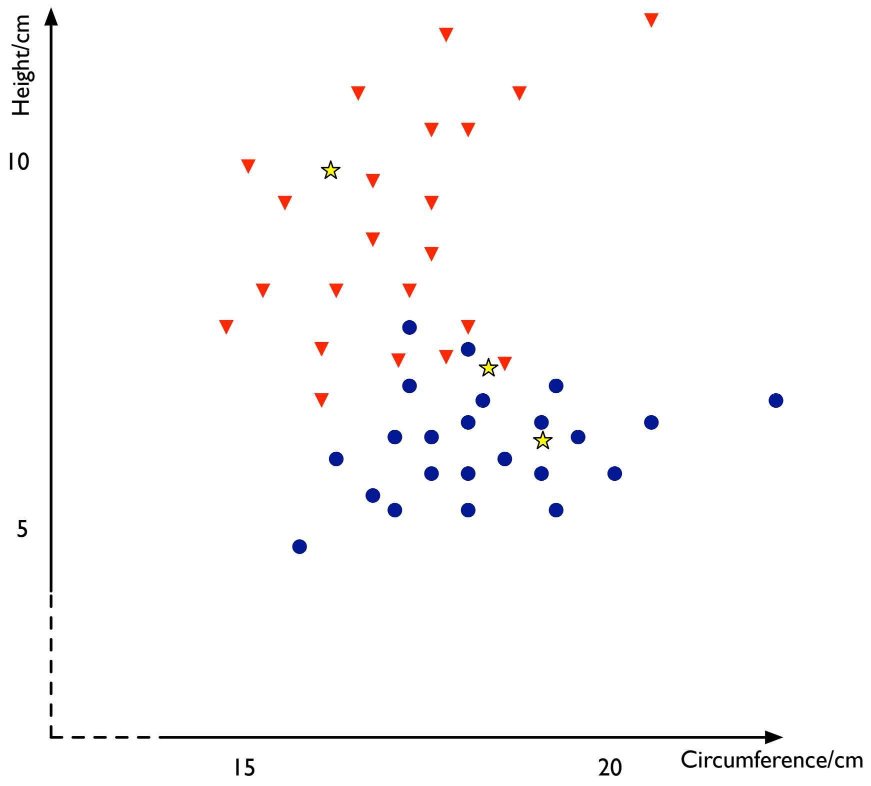


Figure 7.3: The training data for apples (blue circles) and pears (red triangles), together with three test points (yellow stars).

Constructing classifiers using supervised learning To construct the classifier automatically we need:

1. a set of training data containing feature vectors and their labels
2. an algorithm that we train using the training data
3. **hyperparameters**, numbers that control how the algorithm learns and predicts

This is referred to as **supervised learning**, since the label for a training vector acts as supervision for the classifier when it is learning from training data.

7.2 Nearest neighbour classification

Principle of nearest neighbour classification Nearest neighbour classification (or one-nearest neighbour classification to be precise) has a very simple basis: to classify a test item, find the item in the training set which is closest and assign the test item to the same class as the selected training item. If there happens to be an identical item in the training set then it makes sense to assign the test item to the same class. Otherwise, the class of the member in the training set which is most similar to the test item is our best guess. We use a distance measure (e.g., Euclidean distance) to determine similarity. If we have a representation for which the distance measure is a reasonable measure of similarity, then the nearest neighbour method will work well.

Decision boundaries for nearest neighbour classification What do the decision boundaries look like for nearest neighbour classification? Each training data point defines a region around it; test points within this region will be classified to the same class as the training data point. These regions are illustrated for a simple case in Figure 7.4, where the boundaries of regions are shown as dotted lines. Each boundary is given as the perpendicular bisector of the line segment between the two corresponding data points. This partitioning formed by a set of data points is sometimes called **Voronoi diagram** or **Voronoi tessellation**.

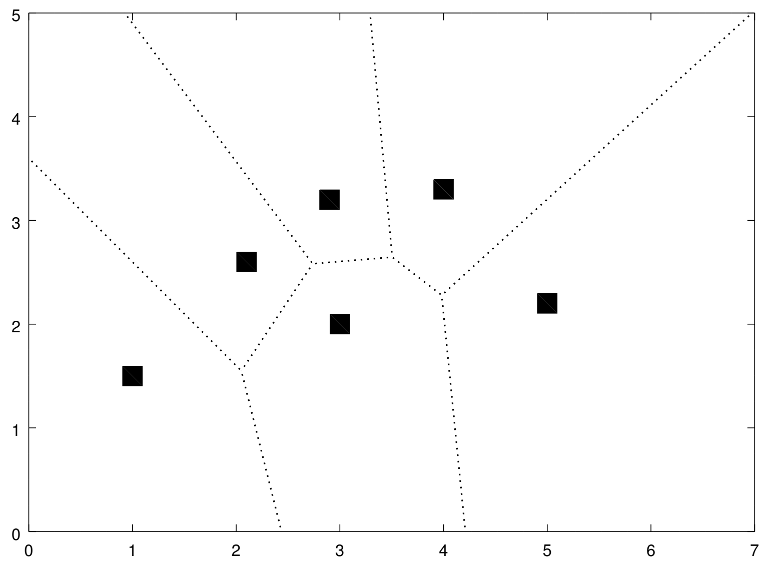


Figure 7.4: Decision boundaries by 1-NN for data points of distinct classes from each other

Now we assume that each data point belongs to either of two classes, say, red or blue. To obtain the decision boundary we combine those boundaries which are between regions of different classes, as illustrated in Figure 7.5. The resultant boundary is referred to as being **piecewise linear**. Figure 7.6 shows the decision boundary and decision regions in the case of three classes.

Application of 1-nearest neighbour to a real dataset Figure 7.7 shows 1 nearest-neighbour classification applied to the fruit dataset. Note that we've standardised the variables, so that the data spreads out roughly equally in both directions. In common with other distance-based methods, we would like the results of clustering to be independent of the units we measure the variables in. It can be seen that the decision boundary is quite complex, with islands of apple amongst the oranges. We'll explore in the next section if this might be a problem.

7.3 Evaluation

Classification error rate Having constructed a classifier, we would like to evaluate how well it works. One way to quantify how well a classifier is working is the number of items that it misclassifies – i.e., the number of times it assigns a class label $\hat{c}_i$ different from the true class label c_i. The classification error is often expressed as the percentage of the total number of items that is misclassified, the **error rate**.

Error rate for one-nearest neighbour classification For one-nearest neighbour classification, the error rate when we consider members of the training set is 0, since the closest point in the training set to a member of the training set is itself[1].

Evaluating generalisation to unseen data This sounds very promising, until we remember that the job of the classifier is to classify data points that we haven't seen before. It may be that the classifier will not **generalise** to data that haven't seen. In order to estimate how well the classifier generalises, we can split our original data

[1]Unless we have two data points with exactly the same features and different labels.

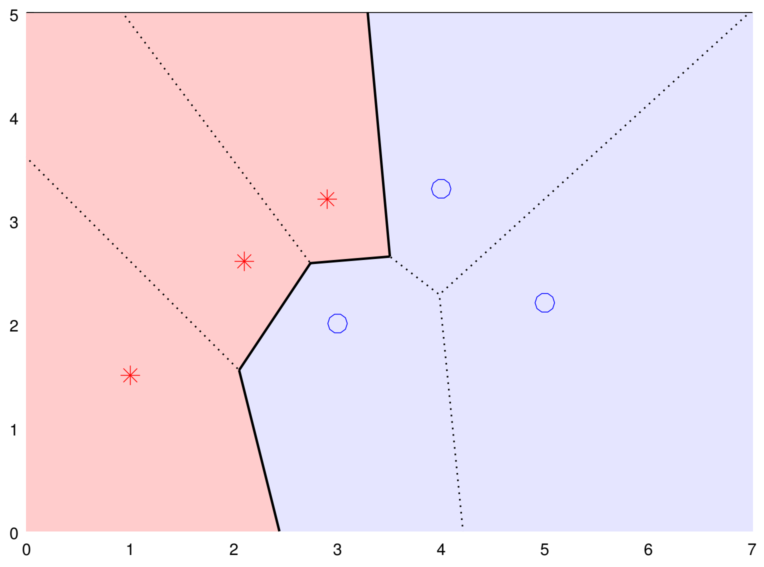


Figure 7.5: Decision boundary and decision regions for a 1-nearest neighbour classifier for a training data set of two classes, where training samples of one class are shown with '*' in red, those of the other class are shown with '○' in blue. The Euclidean distance is used as the distance measure in this example.

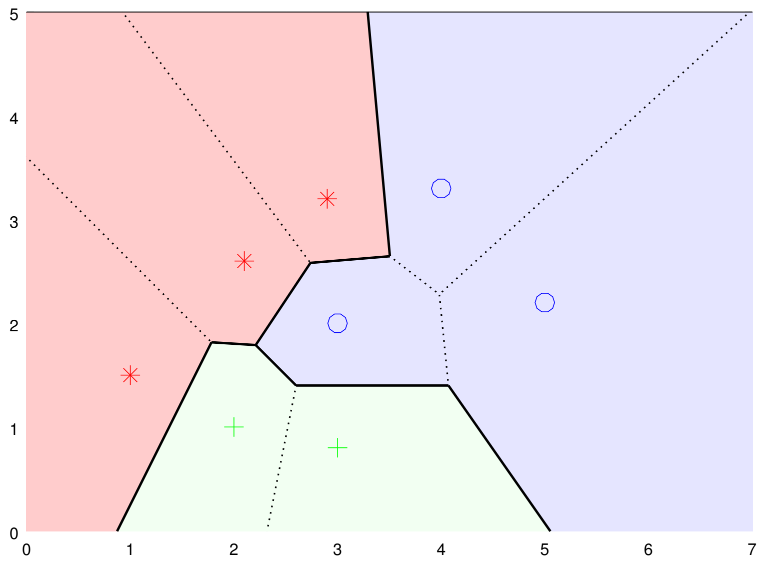


Figure 7.6: Decision boundary and decision regions for a 1-nearest neighbour classifier for three classes.

set into a training set and a **testing set**. The training and testing sets are mutually exclusive, and a typical split

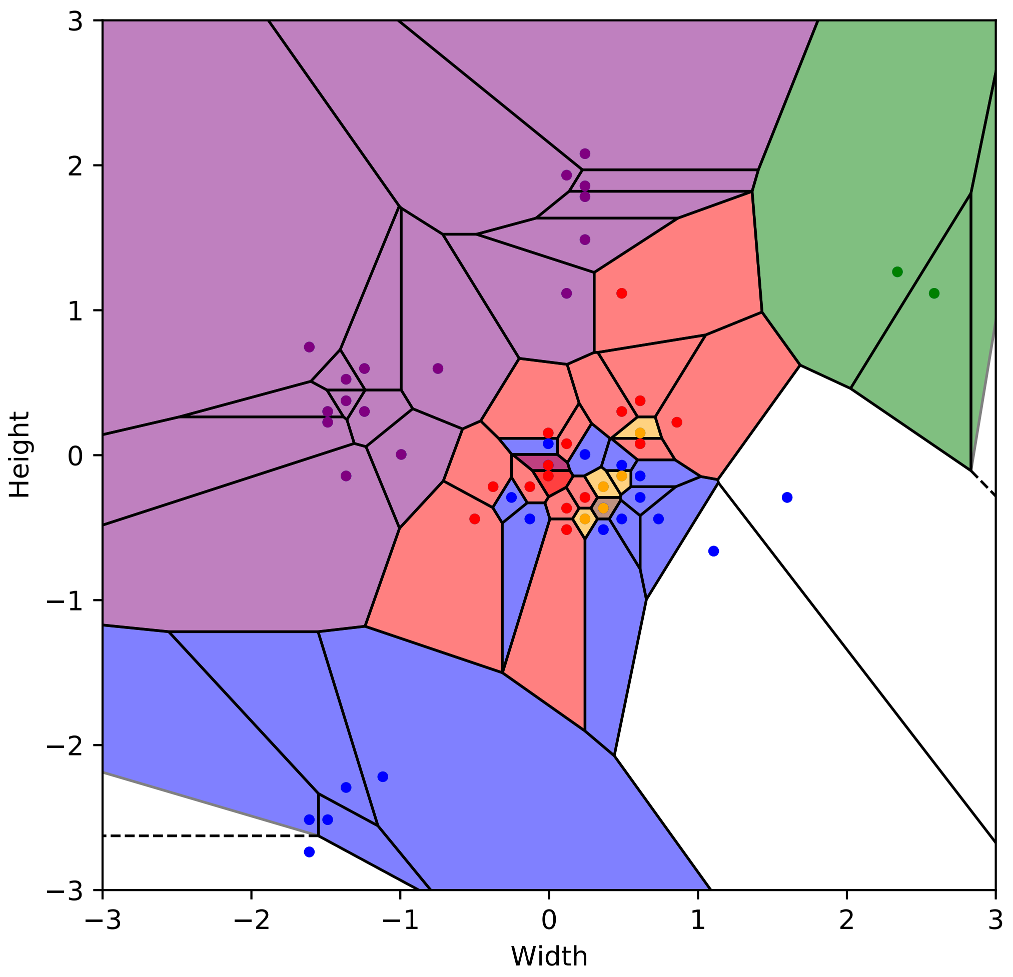


Figure 7.7: Decision regions for one nearest neighbour classification applied to the fruit data set. The variables have been standardised to make the scales on both axes similar. Some regions are darker shade of blue or red. This indicates that there are 2 points labelled with "apple" or "orange" in the dataset with the same features. There is one region that is purple, amongst the blue and red region. There are two data points corresponding to this region with identical coordinates, one labelled with orange and one with apple. Colour indicates decision region for each type of fruit: Apple (blue), Mandarin (orange), Big Orange (green), Orange (red), Lemon (purple).

might be 70% for training and 30% for testing. We train the classifier using the training set, and then evaluate the performance using the testing set. **We are not allowed to use the test set to train the classifier** – otherwise our estimate of its performance on the test set will be too optimistic.

In summary, the **training set error rate** is the percentage of misclassifications that the classifier makes on the training set after the learning algorithm has been applied. The **test set error rate** refers to errors made by the classifier on the testing set.

Training and testing set notation We'll use the notation: $\mathbf{x} = (x^{(1)},\ x^{(2)}, \ldots, x^{(D)})^T$ to denote a D-dimensional (input) feature vector, which has class label c. The **training set** is a set of n feature vectors and their class labels; the i'th training item consists of a feature vector $\mathbf{x}_i$ and its class label c_i. The j'th element of the i'th feature vector is written x_{ij}.

Chapter 8

k-Nearest neighbour classification, setting hyperparameters, and metrics

8.1 k-Nearest neighbour classification

Principle of k-nearest neighbour classification Rather than just using the single closest point, the k-nearest neighbour approach looks at the k points in the training set that are closest to the test point; the test point is classified according to the class to which the majority of the k-nearest neighbours belong.

Figure 8.1 repeats the apples and pears example, with the three test points at $(16, 10)$, $(19, 6)$, and $(18, 7)$. For the first two items, the value of k is not really important: $(16, 10)$ is classified as pear and $(19, 6)$ is classified as apple, no matter how many nearest neighbours are considered. However, the third example above, $(18, 7)$, is ambiguous, and this is reflected in the sensitivity of the classifier to the value of k:

- 1-nearest: classified as pear
- 2-nearest: tie (one apple and one pair are nearest neighbours). In this case, we could choose randomly between the two classes. Another option strategy would be to take the 1-nearest neighbour or the 3-nearest neighbour classification.
- 3-nearest: classified as apple
- 5-nearest: classified as pear
- 9-nearest: classified as apple

k-nearest neighbour algorithm We can write the k-nearest neighbour algorithm precisely as follows, where and d is the distance metric (typically the Euclidean distance):

- For an unseen example $\mathbf{x}$:
  - Compute the n distances $d_i = d(\mathbf{x}, \mathbf{x}_\mathrm{i})$ between $\mathbf{x}$ and the features of each training example $\mathbf{x}_i$, $i \in 1, \dots n$.
  - Sort the distances from lowest to highest and find the indices $i_1, \dots i_k$ of the k lowest values of d_i
  - Find the classes that belong to the closest points, i.e. $c_{i_1}, \dots, c_{i_k}$
  - Each of these represents a vote for a class. Count the votes for each class and return the one with the largest number.
  - If there is a tie, choose randomly, or look at the $k + 1$th neighbour to resolve the tie.

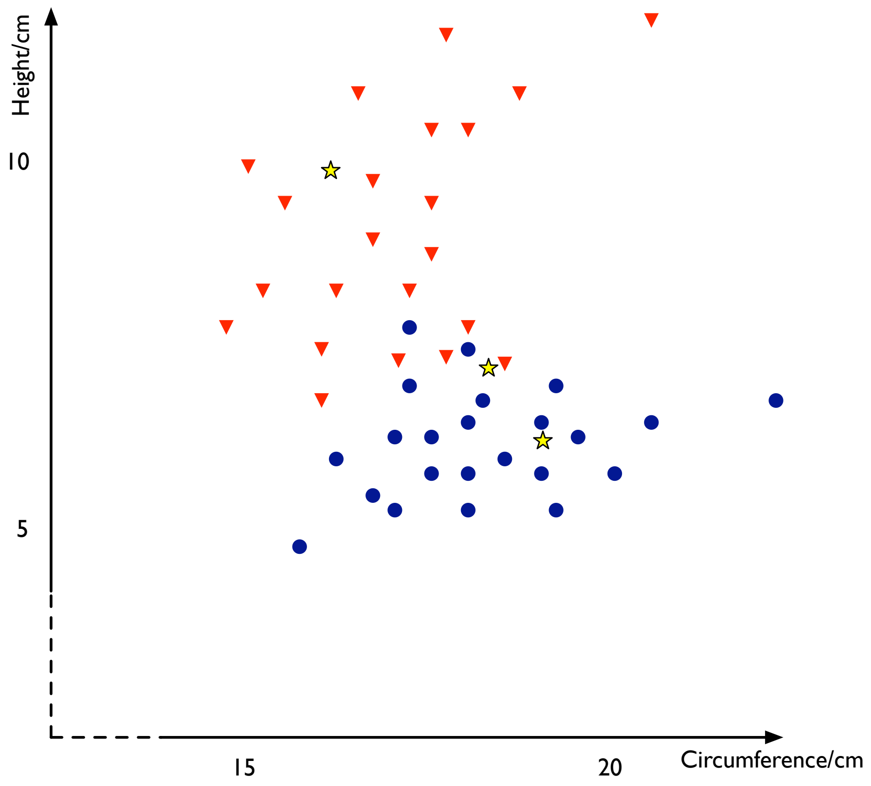


Figure 8.1: The training data for apples (blue circles) and pears (red triangles), together with three test points (yellow stars).

Decision boundaries produced by k-NN classification k-nearest neighbour classifiers make their decisions on the basis of local information. Rather than trying to draw a linear decision boundary across the whole space (as in Figure 7.2), k-nearest neighbour techniques make decisions based on a few local points. As such they can be quite susceptible to noise, especially if k is small: a small shift in the location of a test point can result in a different classification since a new point from the training data set becomes the nearest neighbour. As k becomes larger, the classification decision boundary becomes smoother since several training points contribute to the classification decision. Figure 8.2 illustrates the decision regions for various values of k for the 5-fruit example introduced in the previous chapter (raw data in Figure 7.1 and 1-nearest-neighbour classification in Figure 7.7).

Generalisation and regularisation In Figure 8.2 for low values of k the boundary between apples (blue) and oranges (red) is "noisy": a small shift in the height and width of the fruit will lead to the classification training. The trained classifier is very flexible and therefore over-sensitive to the data it's been training on, and we say that it is exhibiting **over-fitting** and **under-generalisation**.

As k increases, the decision boundaries get smoother, and we might think that the results will be exhibit better **generalisation** to unseen examples. As k increases further there could also the problem of **over-generalisation** or **over-fitting**. This problem isn't seen clearly in Figure 8.2. However, if we made k very large, we would end up classifying everything as the fruit with the largest number of examples. Another example of over-generalisation might be the linear decision boundaries in Figure $^{7.2}$ in Supervised learning: Classification with Nearest neighbours.

We can see that the decision regions with higher k in Figure 8.2 appear more regular. The process of changing the behaviour of a classifier so that it produces more regular or smoother output is known as **regularisation** and k is sometimes referred to as a **regularisation parameter**.

Over- and under-fitting (and their counterparts under- and over-generalisation) are issues for other supervised learning methods, for example when extending multiple regression with extra features (Interaction terms and nonlinear fits). In general, supervised machine learning models have hyperparameters that act as regularisation

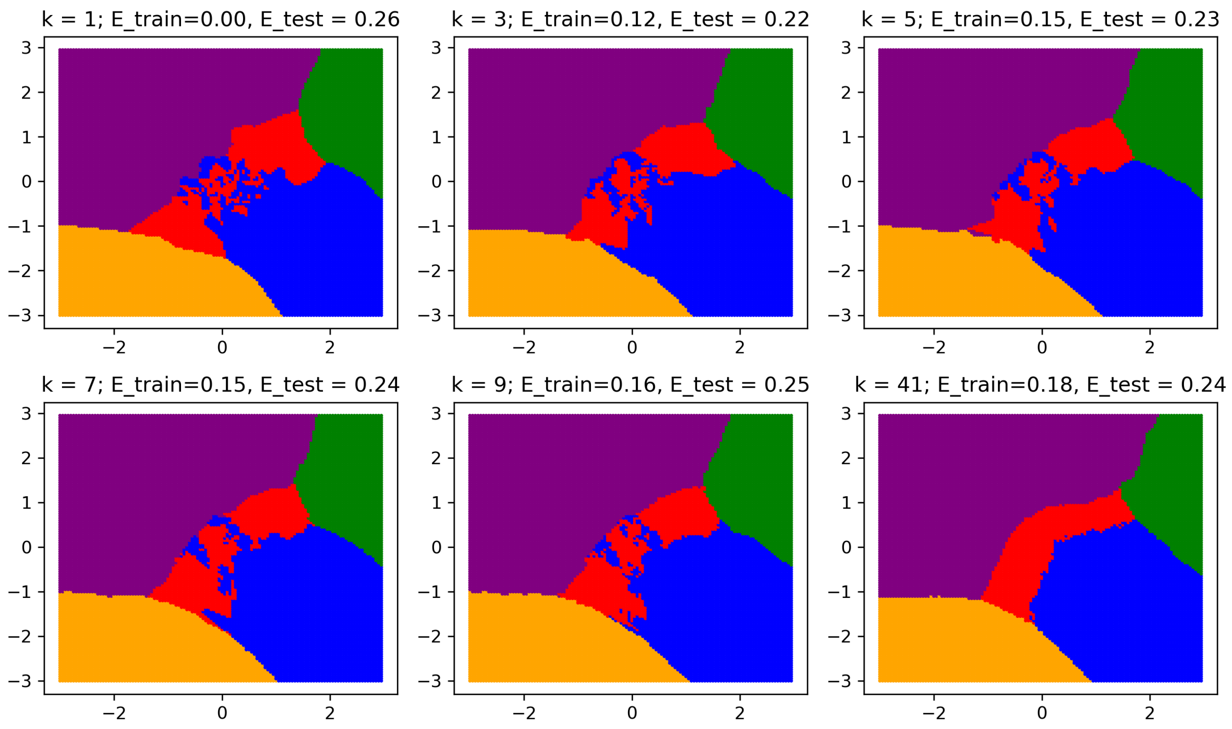


Figure 8.2: Decision regions, training error and testing error for various values of k applied to the 5-fruit data shown in Figure 7.1). Colour indicates decision region for each type of fruit: Apple (blue), Mandarin (orange), Big Orange (green), Orange (red), Lemon (purple).

parameters. We leave more detailed work on regularisation parameters for later courses.

Choosing k The value k is **hyperparameter**: a number that we can choose to get the best performance from the algorithm. Figure 8.3 shows the classification error rate for various values of k on the training set and the testing set. As k increases the error on the training set initially increases rapidly, as explained in the last section. The testing error decreases a little and then starts rising around $k = 9$, indicating that a somewhat larger k helps generalisation. Both testing and training error then increase.

This graph suggests that we can look at the error on the testing set to set k. **But this would break the rule of using the test data to train the classifier, since our choosing the best hyperparameter** k **is part of the training process.** We have really been using the test data **validation data**, that is, data used to help us validate our choice of hyperparameter.

Thus, we need to divide our dataset into 3 parts (Figure:

- Training data (about 50%): used to train the classifier for any particular value of k.
- Validation data (about 25%): used to compare performance of the trained classifier for different values of k.
- Testing data (about 25%): used to **report** the performance of the trained classifier with the one value of k that we have chosen.

The precise fractions of data are not crucial. However, it is important that the testing data is only used to report the performance. A poor test score is probably an indication that the classifier will perform poorly on real world data.

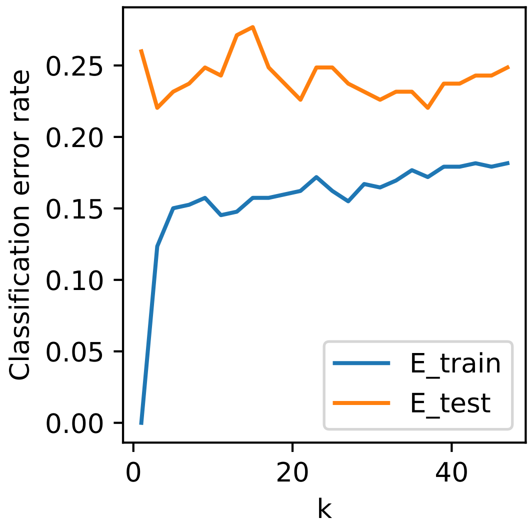


Figure 8.3: Classification error rate for various values of k.

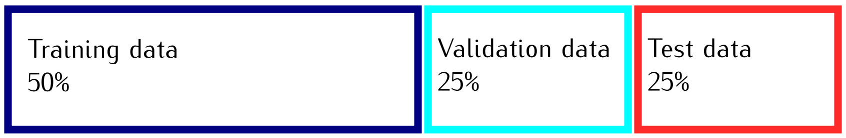


Figure 8.4: Training/validation/testing split.

Computational efficiency of k-nearest neighbour classification k-nearest neighbour is very efficient at training time, since training simply consists of storing the training set.[1] Testing is much slower, since, in the simplest implementation, it would involve measuring the distance between the test point and every training point, which can make k-nearest neighbour impractical for large data sets.

Improving the efficiency of k-NN It is sometimes possible to store the training data set in a data structure that enables the nearest neighbours to be found efficiently, without needing to compare with every training data point (in the average case). One such data structure is the k-d tree. However, these approaches are usually only fast in practice with fairly low-dimensional feature vectors, or if approximations are made.

8.2 Metrics

Accuracy In the chapter on Supervised learning: Classification with Nearest neighbours, we introduced the error rate, the number of items misclassified as a fraction of the total number of items. We define **classification accuracy** as one minus the error rate, i.e. one minus the number of items misclassified divided by the number of items. When the error rate is zero, the classification accuracy is 1 or, equivalently, 100%.

Accuracy and unbalanced classes Accuracy seems to make sense as a metric – the fewer errors, the better the classifier. However, it can appear misleadingly high when there is a large difference in the number of items in each class, which we call **unbalanced classes**. For example, suppose we have a data set containing 95% apples, 3% pears and 2% oranges. When we split into training and test sets, the split will be 95%/3%/2% in both the training and test sets. We could devise a classifier that classifies *any* item as an apple, regardless of its height and width. This classifier would have an accuracy of 95%. This type of dummy classifier is called a **baseline classifier**, since its performance sets a baseline against which to compare more "intelligent" classifiers.

Sensitivity and selectivity as alternative metrics in two-class problems Classification problems often have two classes, for example someone has or has not repaid a loan, or someone does or doesn't have an illness. In these two-class problems, we regard one class as the "positive" outcome and the other as the "negative" outcome. Confusingly the "positive" outcome is usually the case that we are searching for, which may often be a negative thing – think of testing "positive" for Covid-19.

Two-class problems allow us to introduce other metrics, or sometimes pairs of metrics, that avoid the misleading impression given by accuracy with unbalanced classes. One common pair of metrics are **sensitivity** and **selectivity**, defined as:

Sensitivity Fraction of positives classified as positive

Selectivity Fraction of negatives classified as negative

Suppose now that instead of classifying fruit, we are classifying whether someone is "positive" for Covid, on the basis of their symptoms. We will assume that 2% of people truly have Covid. The dummy classifier described has a high accuracy (how high?). However, it classifies all the positive cases as negative, so the sensitivity is 0%. Conversely, all the negative cases are classified as negative, so the selectivity is 100%. This gives us a much clearer picture of the performance of the classifier than accuracy alone. An ideal classifier would have 100% selectivity and sensitivity.

Confusion matrix The fullest picture of the performance of a classifier on a two-class problem can be gained by looking at the confusion matrix, which compares the actual class and the predicted class (Figure 8.5). The cells of the matrix are the number of items classified as "true positives", "false positive" etc. Sometimes we normalise by dividing every cell by the total number of items, so that the sum of all the cells is 100%.

We can define metrics in terms of the cells of the confusion matrix. For example, we have

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{Selectivity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{8.1}$$

[1] In practice, responsible machine learning practitioners will try out different choices of k, and different distance measures, possibly optimising free parameters of a distance measure. Then training requires testing different choices, and becomes expensive.

		Predicted class	
		P	N
Actual	P	TP	FN
class	N	FP	TN

Figure 8.5: Confusion matrix. TP – number of true positives; FN – number of false negatives; FP – number of false positives; and TN – number of true negatives.

Other metrics can be constructed from the numbers in the cells of the confusion matrix; the appropriate metric to use will depend on the application.

8.3 Limitations in supervised machine learning and cross-validation

Limitations in and pitfalls of supervised machine learning The supervised machine learning paradigm can be summarised as:

1. Select algorithm
2. Split data into training, validation and test sets
3. Use training and validation data to select a hyperparameter
4. Report performance of resulting classifier

We've used k-nearest neighbours as our algorithm, but many other algorithms are available.

There are a number of issues when using this paradigm in the real world.

Data can change over time. For example typical Covid symptoms changed over the pandemic, so a classifier that worked well at the start of the pandemic wouldn't necessarily work well at the end of the pandemic. Perhaps it's been a particularly good year for apples in the year we've trained the fruit classifier, and they are larger then in other years.

Selection of training, validation and test sets A related problem is that the training, validation and test sets should be drawn from the same distribution. The training, validation and test sets should be selected using random sampling, As discussed in the chapter on Randomness, sampling and simulation, any form of non-random sample (for example taking the first 50% of data collected over time) could lead to the statistics of the training and test sets being different.

Limited amount of data Machine learning algorithms need more data to perform well. If you have 100 data points, there will be only 50 data points to use for training, 25 for validation and 25 for testing. As discussed in the chapter on Randomness, sampling and simulation, these samples may not be representative of the wider population.

Cross-validation We've seen how we can split the data we're not using to report the final test result into training and validation data, and that this can be used to find the hyperparameter that gives the best performance. However, if we've a small amount of data, we're only using a third of the actual data that we're allowed to use to test the performance of each hyperparameter. The method of **cross-validation** allows us to use all the non-testing data for both training the classifier with each hyperparameter, and testing it.

Figure 8.6 gives an overview of how cross-validation works. There is a split into test data, and non-testing data, as previously. The difference is that the non-testing data is split into a number (here 3) of equally-sized **folds**, or blocks. For each value of the hyperparameter, the model is trained 3 times: first on the data in folds 2 and 3, then on the data in folds 1 and 3, and finally on the data in folds 1 and 2. Each trained model is tested on the data in the fold that wasn't used for training. The mean error from all three folds is then used as a *estimate* of what the test error will be in the final trained model. On the basis of this cross-validation, a hyperparameter is chosen – probably the one giving the lowest mean error. Finally, the classifier is trained again using the chosen hyperparameter on all the non-testing data.

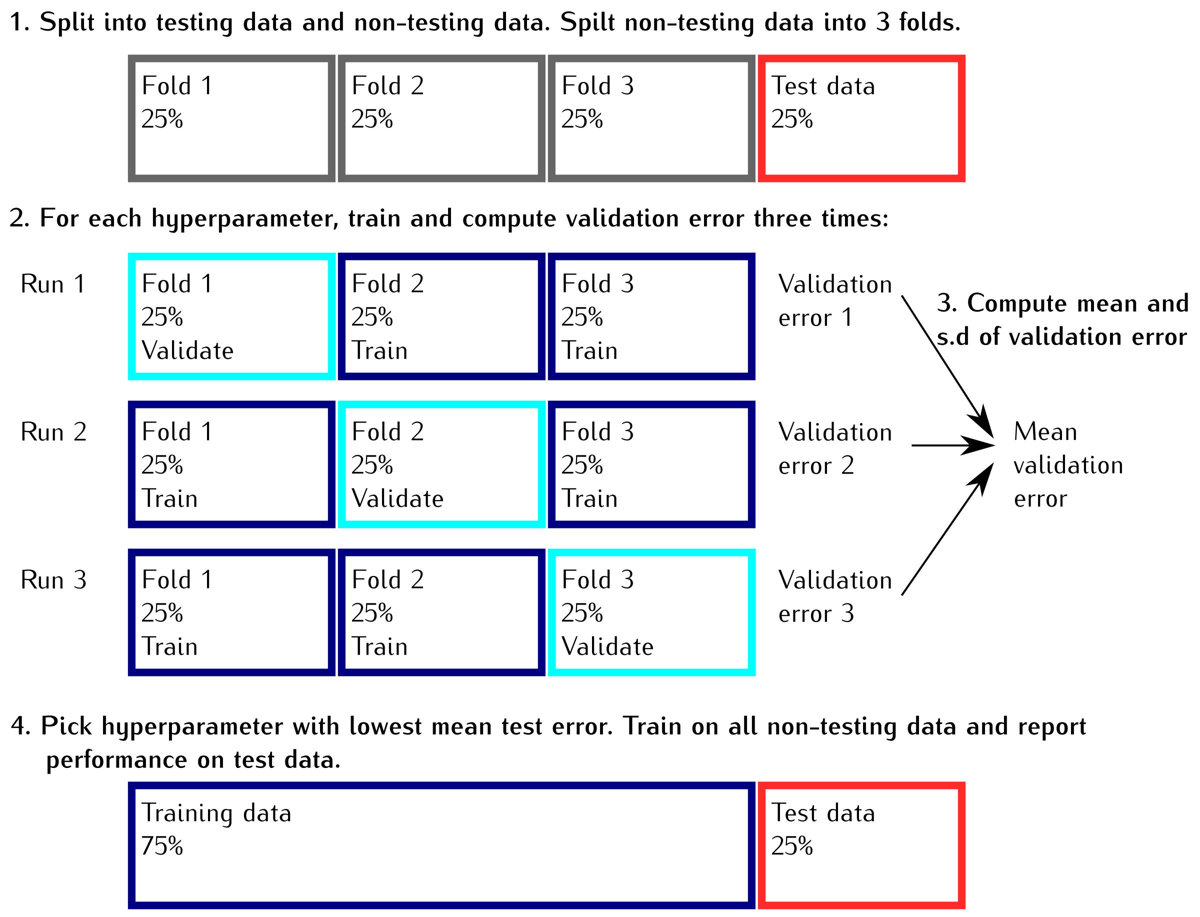

1. Split into testing data and non-testing data. Spilt non-testing data into 3 folds.
Fold 1 25%
Fold 2 25%
Fold 3 25%
Test data 25%
2. For each hyperparameter, train and compute validation error three times:
Run 1
Fold 1 25% Validate
Fold 2 25% Train
Fold 3 25% Train
Validation error 1
Run 2
Fold 1 25% Train
Fold 2 25% Validate
Fold 3 25% Train
Validation error 2
Run 3
Fold 1 25% Train
Fold 2 25% Train
Fold 3 25% Validate
Validation error 3
3. Compute mean and s.d of validation error
Mean validation error
4. Pick hyperparameter with lowest mean test error. Train on all non-testing data and report performance on test data.
Training data 75%
Test data 25%

Figure 8.6: Cross-validation. See text for details.

In general, we don't have to have 3 folds. We can have a number k of folds, which gives rise to the term k-fold cross-validation. 5 and 10 are common values for k. It's unfortunate that the letter k is used for both cross-validation and nearest neighbours, as k means different things in the two contexts.

Although the validation error gives an estimate of the expected test error, it is still necessary to compute the error on a separate test set. The estimate produced by cross-validation has an optimism bias, because the data in each testing fold has been used to select the model hyperparameters.

However, in one case it is OK to used all the data for training and cross-validation: if someone else is doing the testing using a part of the data that they have retained and shared with you – this is common in machine learning competitions. They will get the independent estimate, and you can use the data that you have to best train your models.

Related Python Lab: k-nearest neighbours

`https://github.com/Inf2-FDS/FDS-S1-09-knn`

Chapter 9

Unsupervised learning: K-means

Further reading (not examinable)

- MacKay, D. J. C. (2003) *Information Theory, Inference and Learning Algorithms* pp. 284-288
- Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165. `https://doi.org/10.1007/s40745-015-0040-1`

9.1 Clustering, unsupervised and supervised learning

Cluster analysis aims to partition a data set into meaningful or useful groups, based on distances between data points. In some cases the aim of cluster analysis is to obtain greater understanding of the data, and it is hoped that the clusters capture the natural structure of the data. In other cases cluster analysis does not necessarily add to understanding of the data, but enables it to be processed more efficiently.

As David MacKay put it:

> Human brains are good at finding regularities in data. One way of expressing regularity is to put a set of objects into groups that are similar to each other. For example, biologists have found that most objects in the natural world fall into one of two categories: things that are brown and run away, and things that are green and don't run away. The first group they call animals, and the second, plants. (MacKay, 2003), p. 284

Our aim is to get algorithms to do what human brains do naturally.

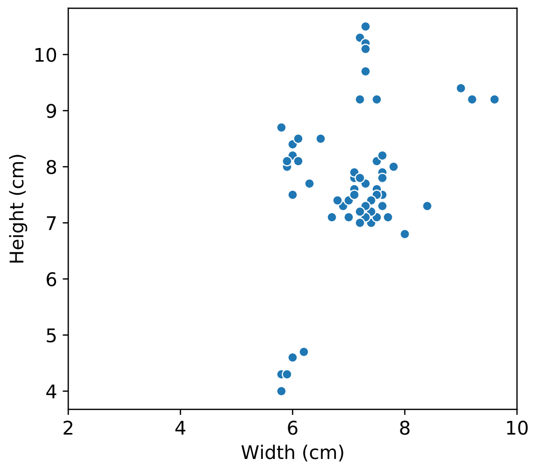
Supervised learning Cluster analysis can be contrasted with **classification** (Figure 9.1). Classification is a **supervised** learning process: there is a training set in which each data item has a label. For example, a bank might want to predict whether a customer might be able pay back a bank loan on the basis of variables such as their income, number of loans paid back, housing status, employment status. The training set is the equivalent data for previous customers, labelled with whether they did or didn't pay back their loans. In the test set the label for each customer is unknown: it is the job of the classifier to predict a label for each test item.

Unsupervised learning Clustering, on the other hand, is an **unsupervised** learning process in which the training set does not contain any labels. The aim of a clustering algorithm is to group such a data set into clusters, based on the unlabelled data alone. In many situations there is no 'true' set of clusters. For example consider the twenty data points shown in Figure 9.2 (a). It is reasonable to divide this set into two clusters (Figure 9.2 (b)), four clusters (Figure 9.2 (c)) or five clusters (Figure 9.2 (d)).

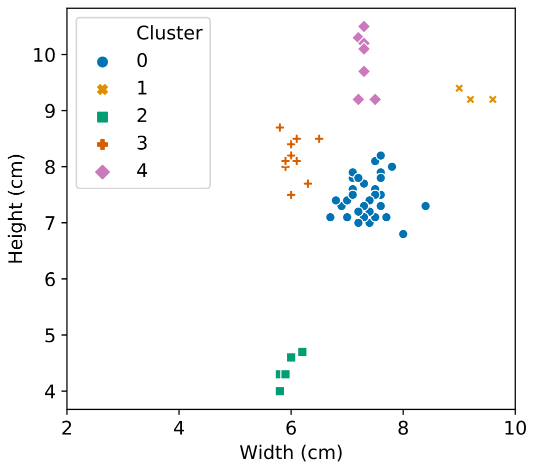
Clustering – a type of unsupervised learning procedure

Input: unlabelled data points. e.g. widths and heights of various unknown fruits

Output: each point is assigned to a cluster – which may correspond to the original fruits



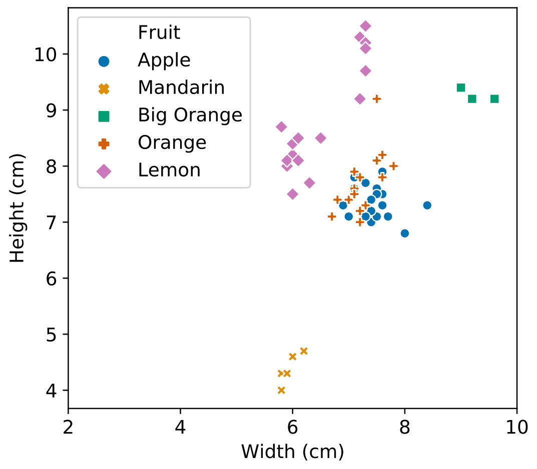
→



Classification – a type of supervised learning procedure

Input: labelled data points. e.g. widths and heights of various fruits

Output: classifier that can predict the identity of an unlabelled data point



→

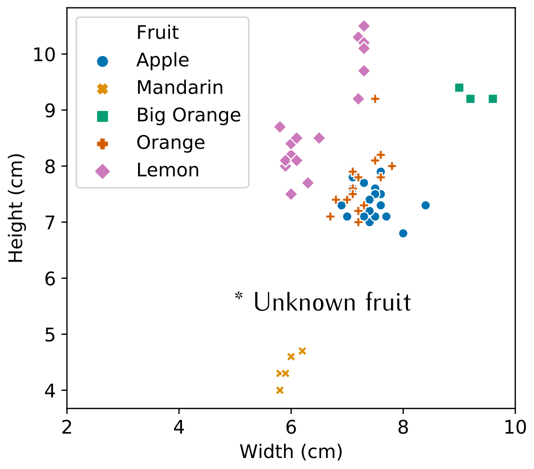
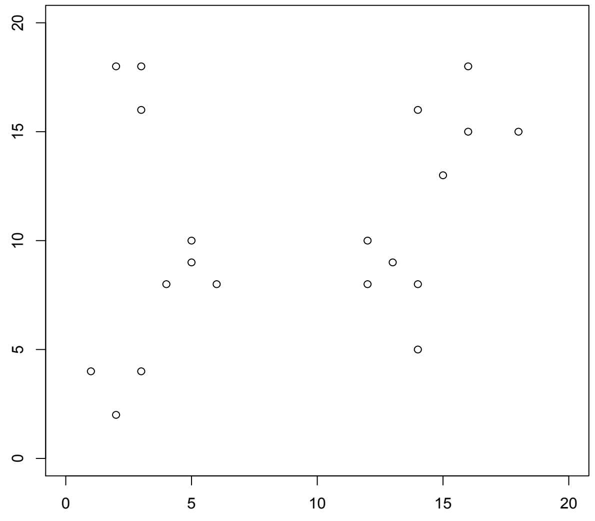
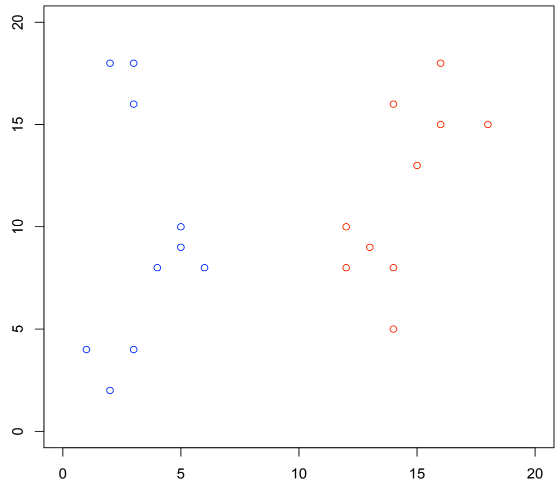
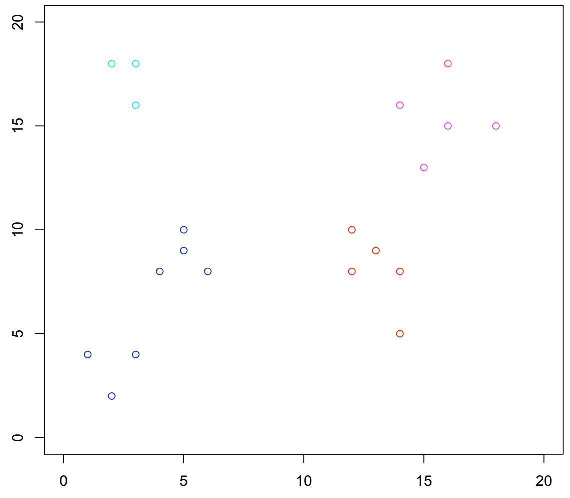


Figure 9.1: Unsupervised and supervised learning, exemplified by clustering and classification applied to Iain Murray's *oranges and lemons* dataset of the widths, heights and masses of various types of fruits.


(a) original data


(b) two clusters


(c) four clusters

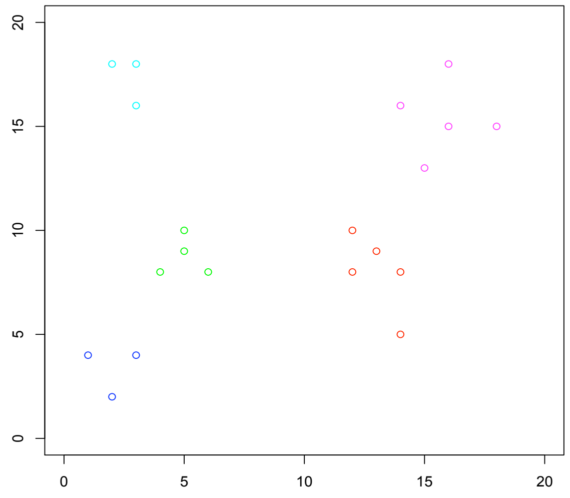

(d) five clusters

Figure 9.2: Clustering a set of 20 two-dimensional data points.

Reasons to cluster There are many reasons to perform clustering. Most commonly it is done to better understand the data (**data interpretation**), or to efficiently encode the data set (**data compression**).

Data interpretation: Automatically dividing a set of data items into groups is an important way to analyse and describe it. Automatic clustering has been used to cluster documents (such as web pages), user preference data, and many forms of scientific observational data in fields ranging from astronomy to psychology to biology.

Data compression: Clustering may be used to compress data by representing each data item in a cluster by a single cluster **prototype**, typically at the centre of the cluster. Consider D-dimensional data which has been clustered into K clusters. Rather than representing a data item as a D-dimensional vector, we could store just its cluster index (an integer from 1 to K). This representation, known as **vector quantisation**, reduces the required storage for a large data set at the cost of some information loss. Vector quantisation is used in image, video and audio compression.

9.2 Types of clustering

Types of clustering There are two main approaches to clustering: hierarchical and partitional. **Partitional clustering** does not have a nested or hierarchical structure. It simply divides the data set into a fixed number of non-overlapping clusters, with each data point assigned to exactly one cluster. The most commonly employed partitional clustering algorithm, K-means clustering, is discussed in the next section. **Hierarchical clustering** forms a tree of nested clusters in which, at each level in the tree, a cluster is the union of its children (Figure 9.3). Whatever approach to clustering is employed, the core operations are distance computations: computing the distance between two data points, between a data point and a cluster prototype, or between two cluster prototypes.

Hierarchical clustering There are two main approaches to hierarchical clustering. In **top-down clustering** algorithms, all the data points are initially collected in a single top-level cluster. This cluster is then split into two (or more) sub-clusters, and each these sub-clusters is further split. The algorithm continues to build a tree structure in a top-down fashion, until the leaves of the tree contain individual data points. An alternative approach is **agglomerative hierarchical clustering**, which acts in a bottom-up way. An agglomerative clustering algorithm starts with each data point defining a one-element cluster. Such an algorithm operates by repeatedly merging the two closest clusters until a single cluster is obtained.

9.3 K-means

Aim K-means clustering aims to divide a set of D-dimensional data points into K clusters. The number of clusters, K, must be specified; it is not determined by the clustering: thus it will always attempt to find K clusters in the data, whether they really exist or not.

Algorithm Each cluster is defined by its cluster centre, and clustering proceeds by assigning each of the input data points to the cluster with the closest centre, using a Euclidean distance metric. The centre of each cluster is then re-estimated as the *centroid* of the points assigned to it. The process is then iterated. The algorithm, illustrated in Figure 9.4, is:

- *Initialise* K cluster centres, $\{\mathbf{m}_k\}_1^K$
- While not *converged*:
  - Assign each data vector $\mathbf{x}_i$ $(1 \leq i \leq n)$ to the closest cluster centre;
  - Recompute each cluster mean as the mean of the vectors assigned to that cluster

To be more precise, we can express the assignment of a point i to a cluster k by defining the set of points in cluster k as $\mathcal{C}_k$. Using the notation $|\mathcal{C}_k|$ to indicate the number of points assigned to cluster k, we can then recompute the mean of cluster k using the following equation:

$$\mathbf{m}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i \tag{9.1}$$

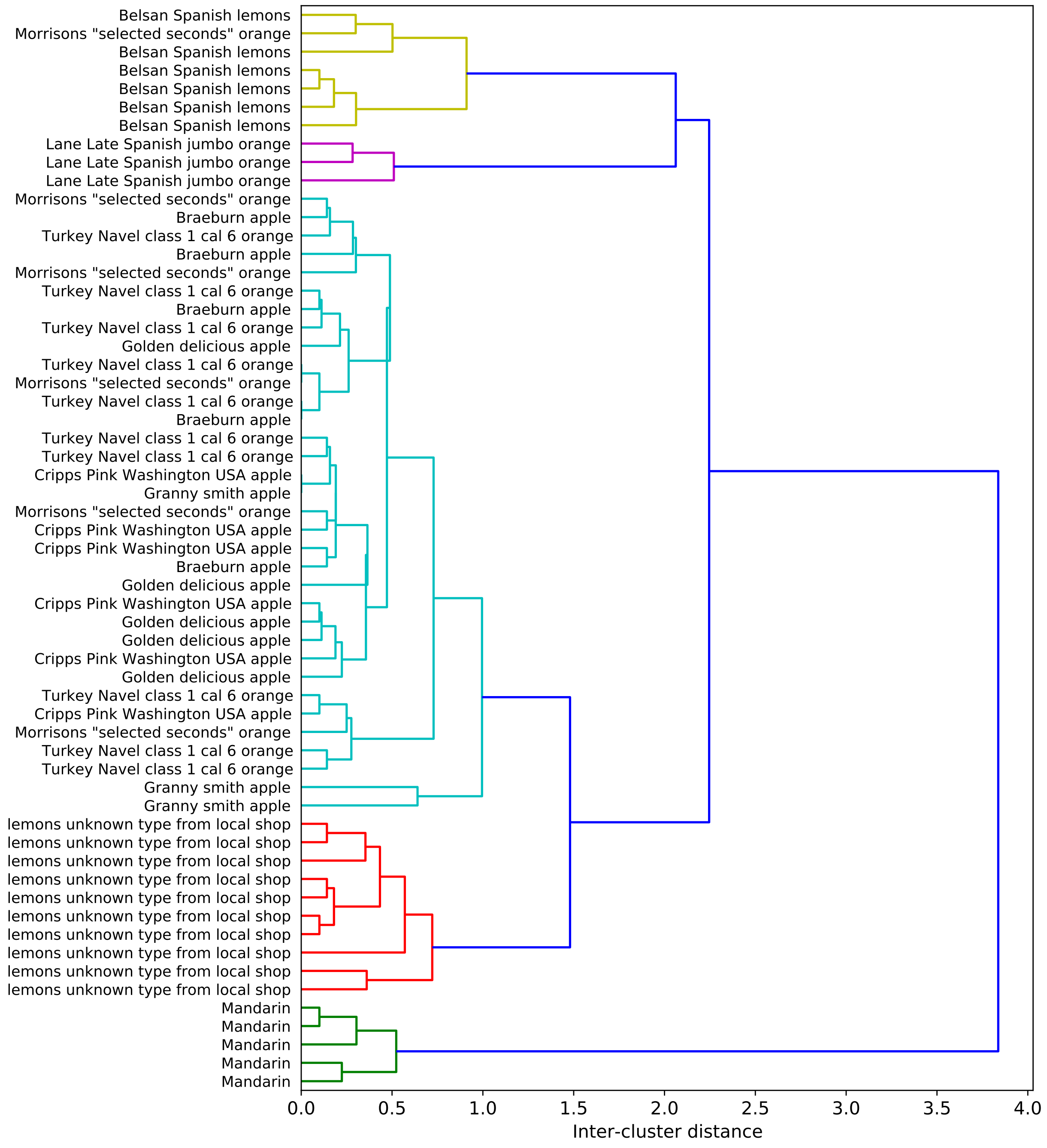
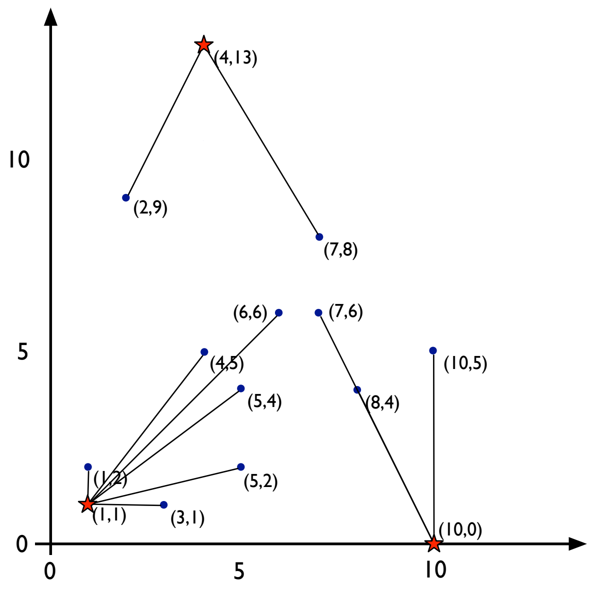
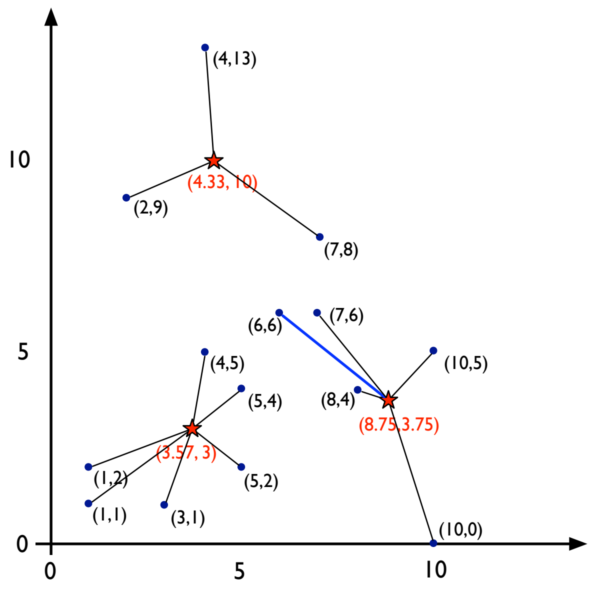


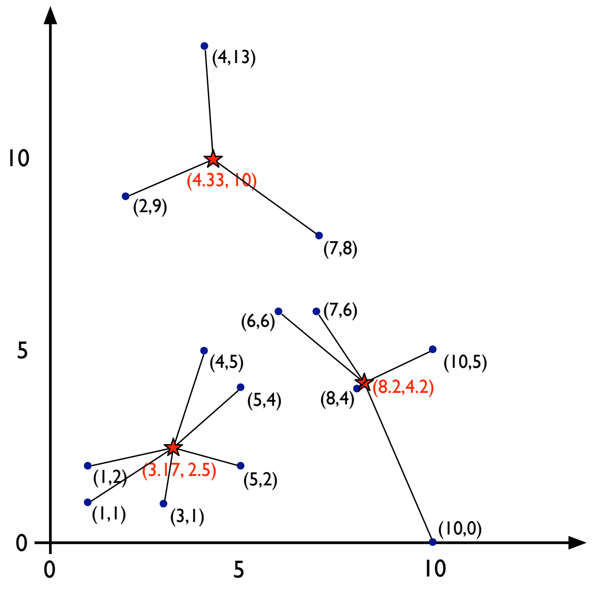
Figure 9.3: Hierarchical clustering of the fruit dataset. The x-axis indicates the distance between clusters, which in this clustering example is the centroid of all the points belonging to the cluster. For example the two Mandarins at the very bottom are about 0.2 units apart, and form a cluster. In turn the centre of this cluster is about 0.45 units from the centre of the next closest cluster (the three other Mandarins). We can regard all sub-trees lower than a threshold inter-cluster distance of our choosing as clusters. For example, if we chose a threshold of 1.0, we would see 5 clusters, indicated by the green, red, cyan, magenta and yellow sub-trees.



(a) Initialisation



(b) After one iteration



(c) After two iterations

Figure 9.4: Example of K-means algorithm applied to 14 data points, $K = 3$. The lines indicate the distances from each point to the centre of the cluster to which it is assigned. Here only one point (6,6) moves cluster after updating the means. In general, multiple points can be reassigned after each update of the centre positions.

Distance measure The algorithm requires a distance measure to be defined in the data space, and the Euclidean distance is often used. In this case, if $\mathbf{x}$ and $\mathbf{y}$ are two points in the data space, the distance function $d(\mathbf{x}, \mathbf{y})$ is given:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{j=1}^{D} (x_j - y_j)^2} \tag{9.2}$$

where $\| \cdot \|$ denotes the Euclidean norm (i.e. L^2-norm) of a vector.

Initialisation The initialisation method needs to be further specified. There are several possible ways to initialise the cluster centres, including:

- Choose random data points as cluster centres
- Randomly assign data points to K clusters and compute means as initial centres
- Choose data points with extreme values
- Find the mean for the whole data set then perturb into K means

All of these work reasonably, and there is no 'best' way. However, as discussed below, the initialisation has an effect on the final clustering: different initialisations lead to different cluster solutions.

Convergence The algorithm iterates until it converges. Convergence is reached when the assignment of points to clusters does not change after an iteration. An attractive feature of K-means is that convergence is guaranteed. However, the number of iterations required to reach convergence is not guaranteed. For large datasets it is often sensible to specify a maximum number of iterations, especially since a good clustering solution is often reached after a few iterations. Figure 9.4 illustrates the K-means clustering process. Figure 9.5 illustrates how different initialisations can lead to different solutions.

9.4 Evaluation and application of K-means clustering

Multiple solutions Depending on the initialisation of the cluster centres, K-means can converge to different solutions; this is illustrated in Figure 9.5 in which the same data set of 4 points has two different clusterings, depending on where the initial cluster centres are placed. Both these solutions are **local minima** of the error function, but they have different error values. For the solution in Figure 9.5a, the error is:

$$E = \frac{((4+4) + (4+4))}{4} = 4\,.$$

The second solution (Figure 9.5b) has a lower error:

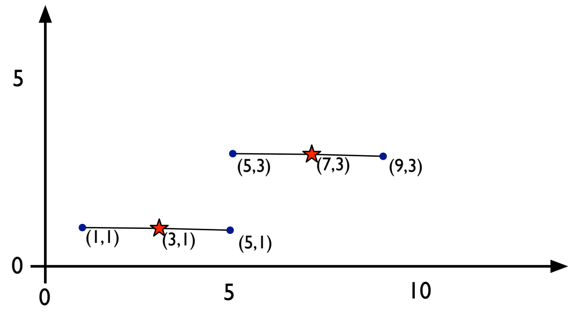
$$E = \frac{0 + (32/9 + 20/9 + 68/9)}{4} = \frac{10}{3} < 4\,.$$

Figures 9.5c and 9.5d show two clusterings of the fruit data set – the first clustering is noticeably worse, with one cluster centre inbetween two distinct groups of points.

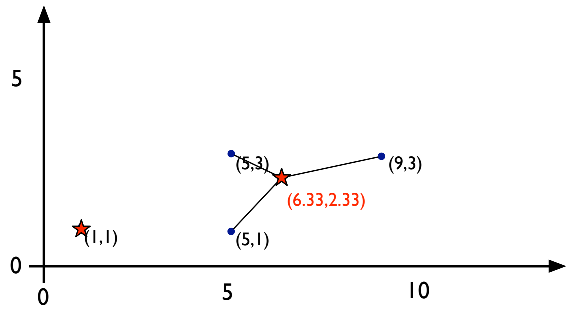
To get around the problem of multiple solutions, it is common to repeat K-means algorithms from multiple randomly-chosen starting points, and then take the solution that gives the lowest value of a mean squared error E, that we will define shortly.

Mean squared error function To compare two different clusterings each with K clusters we can use the **mean squared error** function[1], which can be thought of as measuring the **scatter** of the data points relative to their

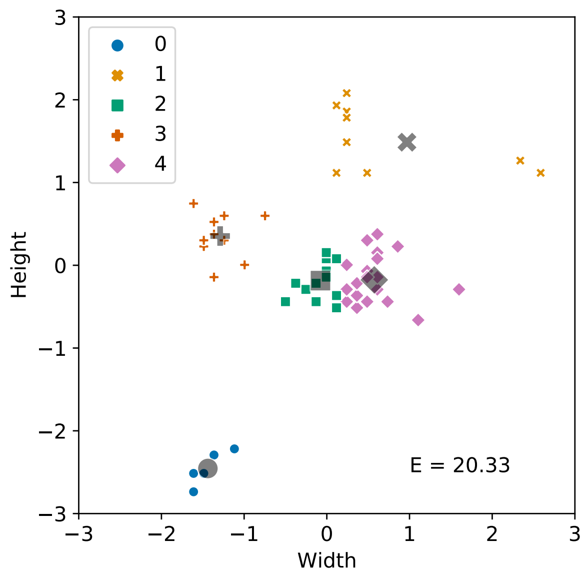
[1]Commonly in K-means, the **sum of the squared errors**, also known as the inertia, is used instead of the mean squared error. It is simply the mean squared error, but without the $1/n$ factor.



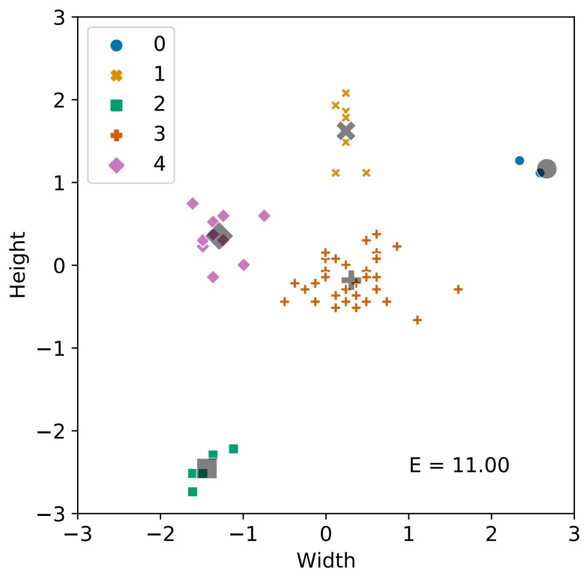
(a) Within-cluster sum-squared error = 4



(b) Within-cluster sum-squared error = 3.33



(c) Within-cluster sum-squared error $E = 20.33$



(d) Within-cluster sum-squared error $E = 11.00$

Figure 9.5: (a) and (b) Two different converged clusterings for the same data set, but starting from different initialisations. (c) and (d) Two different converged clusterings for fruit data set, but starting from different initialisations.

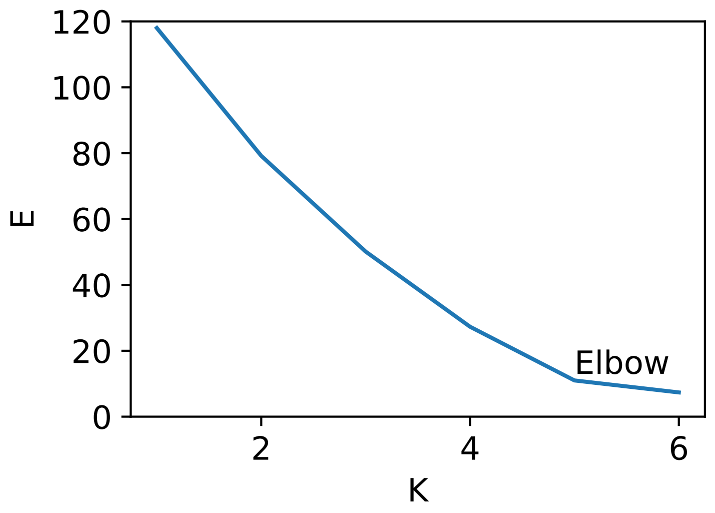


Figure 9.6: Scree plot of K-means clustering applied to the fruit dataset. The elbow is around $K = 5$, implying that 5 is a reasonable number of clusters.

cluster centres. Thus, if we have two sets of K clusters, it makes sense to prefer the one with the smallest mean squared error: the clustering with the lowest scatter of data points relative to their cluster centres.

Let us define the set of points in the kth cluster as $\mathcal{C}_k$, Then we can write the mean squared error as

$$E = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\mathbf{x}_i - \mathbf{m}_k\|^2, \tag{9.3}$$

where $\mathbf{m}_k$ is the centre of cluster k, n is the number of data points in total, and $\| \cdot \|$ denotes the Euclidean norm (i.e. L^2-norm) of a vector.

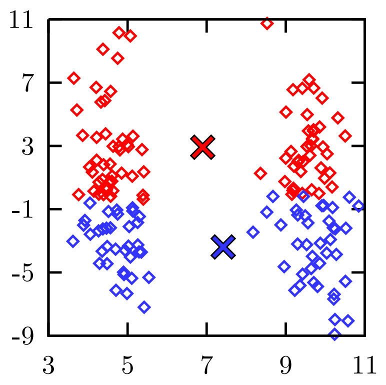
Another way of viewing this error function is in terms of the squared deviation of the data points belonging to a cluster from the cluster centre, summed over all centres. This may be regarded as a variance measure for each cluster, and hence K-means is sometimes referred to as **minimum variance** clustering.

In the introduction we said that the aim of clustering was to discover a structure of clusters such that points in the same group are close to each other, and far from points in other clusters. The mean squared error only addresses the first of these: it does not include a between-clusters term.

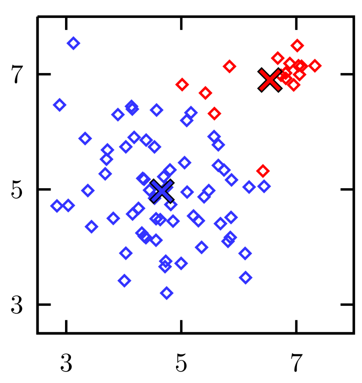
Failures of K means Like all variance-based approaches this criterion is dependent on the scale of the data. This if variables are on different scales, nonsensical clusters can arise (Figure 9.7a). The range of the y-coordinates is around 20, whereas the range on the x-axis is about 8. Thus, a point at the top of the left-hand cluster is closer to a point at the top of the right-hand cluster than it is to a point at the bottom of the left-hand cluster. Standardising variables is often a sensible step to take solve this problem - here it would reduce the distances in the y-direction relative to the x-direction.

Another failure mode of K-means is that large clouds can pull small clusters off-centre (Figure 9.7b). The reason for this is that we've not modelled the variance of each cluster – we've implicitly assumed they are the same. The solution is to allow the variance of the clusters to vary. Gaussian mixture models do this, but are beyond the scope of this course.

How to decide on K Similarly to PCA, we can use a scree plot (Figure 9.6), this time plotting the mean squared error E against the number of clusters K. We are looking for the **elbow**, where the error stops falling dramatically.



(a) Differences in scale on the axes cause nonsensical clusters.



(b) A smaller cloud (red points) attracts points from a larger cloud (blue points).

Figure 9.7: Failures of K-means. Figures by Iain Murray.

Other variants of K-means We have discussed the **batch** version of K-means. The **online** (sample-by-sample) version in which a point is assigned to a cluster and the cluster centre is immediately updated is less vulnerable to local minima.

There are many variants of K-means clustering that have been designed to improve the efficiency and to find lower error solutions.

The curse of dimensionality As the number of dimensions D increases, the distances between points increases. This has the effect of making the intracluster distances on a similar scale to the intercluster distances, and so clustering becomes less reliable. This is one manifestation of the curse of dimensionality. To counteract the curse, a dimensionality reduction such as PCA can be applied to the data before it is clustered.

Related Python Lab: K-means

`https://github.com/Inf2-FDS/week09-kmeans`

Part III

Linear models

Chapter 10

Linear Regression

10.1 Regression as prediction

Regression is related to the correlation coefficient, but is subtly different. In regression, we are still dealing with two numeric variables x and y, but we are trying to predict what value of y will be found at a particular x, not just give a measure of how they are correlated. Often we have good reason to believe that there is a causal relation between the variables x and y such that x causes y or y depends on x.

The variable y can be referred to as the **response variable** (or "response" for short) and the variable x can be referred to as the **predictor variable** (or **predictor** for short). We will use this terminology[1] but it is worth noting that the response variable is also referred to as a **dependent variable**, **target variable**, the **outcome** or **endogenous variable** and the predictor variables are also referred to as **features**, **explanatory variables**, **independent variables**, **covariates**, **regressors**, **exogenous variables**. The terminology used tend to vary on the discipline. In particular, in Machine Learning, the pair "target" and "features" or "covariate" is used; we have used "feature" when introducing Machine Learning in Supervised learning: Classification with Nearest neighbours.

A standard dataset used to illustrate regression involves another mammal, *Homo sapiens*. The data was collected by the inventor of regression, Francis Galton[2], who surveyed the heights of grown-up children and their parents. He expected that parents who were taller than average would have children who were taller than average. Galton's results are shown in Figure 10.1. The data is simplified. Galton recorded four variables: the height of the mother, the height of the father, the gender of the child and the height of the child. We have reduced the number of variables to three by taking the mean height of each child's parents to give a variable we call the midparental height.

Suppose we want to predict the height a daughter will grow to given that we know the heights of her parents, and thus her midparental height. We have a reasonable amount of data, so we could just take the mean height of all daughters with a similar midparental height. To be more specific, if we knew the midparental height were x, we could predict the future height of the daughter to be the mean height of all daughters with midparental heights in the range $[x - 0.5, x + 0.5]$.

Figure 10.2 shows the results of this method of prediction. It's generally true that the taller the parents, the taller the daughters will be, though the relationship is not quite monotonic – it fluctuates a bit at lower midparental heights. It seems reasonable to suppose that, if there were more data, the relationship would be linear. In the next section we'll see what happens if we assume the relationship is linear.

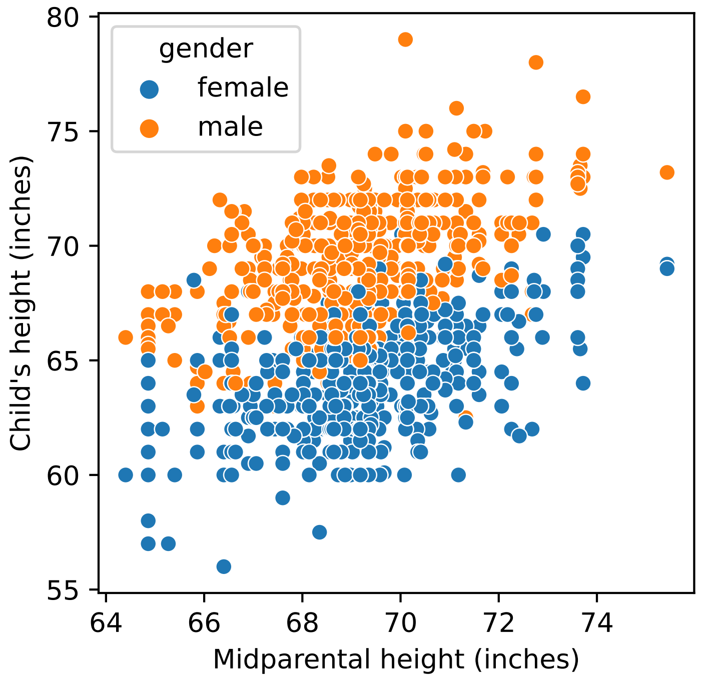


Figure 10.1: Height of daughters and sons plotted against the mean height of their two parents.

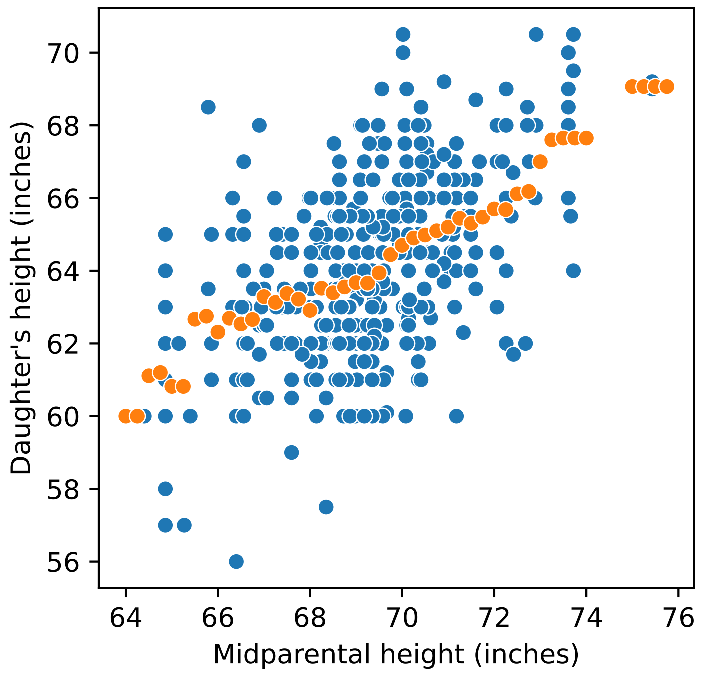


Figure 10.2: Heights of daughters plotted against their midparental heights (blue dots) and predictions (orange dots). The method described in the text doesn't work where there are gaps in the *x* values of greater than 1 inch.

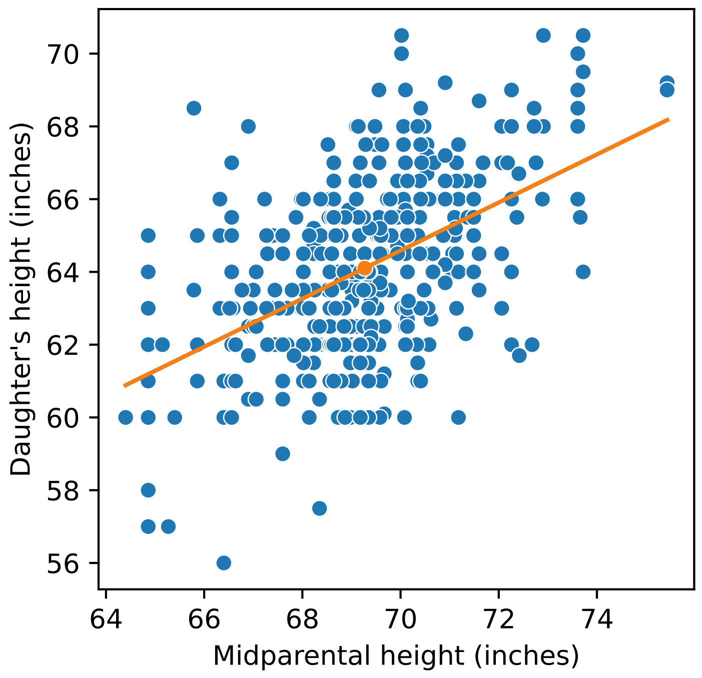


Figure 10.3: Linear regression of daughter's height on their midparental height. The linear regression line is shown in orange and the location of the means of the daughter's height and midparental height is indicated by the orange dot.

Figure 10.4: Left: LEGO Expert Builder set 952. (David Sterratt, CC-BY-4.0) Right: Massey Fergusson 135 Tractor (Lyle Buist, reproduced with permission).

10.2 Linear regression

The linear regression model We'll now assume that y depends linearly on the predictor variable x:

$$y = \beta_0 + \beta_1 x \tag{10.1}$$

The variables β_0 and β_1 are called **parameters** or **regression coefficients** and are the intercept and slope of the line respectively. Think of them as dials that we can turn to produce any straight line we want to. Our aim is to tune the values of β_0 and β_1 so that the line lies close to the data, and will therefore be a good predictor. We refer to Equation 10.1 as the **linear regression model**.

Models The word "model" recurs throughout data science and statistics, and indeed, science in general. A general definition is that a **model** is an abstraction of reality that captures aspects that are important for a given task and omits the rest.

For example, a LEGO® model of a tractor looks like a tractor but doesn't include all the details of a real tractor (Figure 10.4). Similarly, the linear regression model looks like the data, but doesn't contain all the details. Figure 10.3 shows the linear regression model of the data that we are working towards in the next few pages.

Note that linear regression is called a **parametric model**, since it contains explicit parameters (the regression coefficients). In contrast, the method shown in Figure 10.2 is a **non-parametric method**, since the prediction does not depend explicitly on any parameters.

Types of model

The LEGO® tractor is an example of a physical or mechanical model. If we're using equations to connect how gravity and Newton's laws lead to the motion of a pendulum, we'd be using a mathematical model; if we were simulating a city in a computer simulation game, we'd be using a computational model. Biologists refer to animal models, i.e. investigate biological processes such as diseases in an animal, with the aim of understanding these biological process in humans.

The linear regression model is an example of a **statistical model**. We use statistical models to investigate relationships in data and to make predictions. Statistical models can help us to explain the data, but they do not provide a mechanistic explanation of the data. For example, fitting a sine wave to the motion of a pendulum would be a statistical model; in a mathematical model of the pendulum the sine wave would emerge from solving equations describing gravity, Newton's laws, and the weight and length of the

[1] Up until 2022–23, we used the terms "dependent variable" and "independent variable" in FDS. However, the term "independent variable" is confusing, since in multiple regression, the predictor variables are often not statistically independent.

[2] Francis Galton (1822–1911) was a remarkable polymath – and a eugenicist. Adam Rutherford's book *Control: the Dark History and Troubling Present of Eugenics* (Rutherford, 2022) gives a detailed account of Galton's views and those of his academic offspring and fellow statisticians and eugenicists Karl Pearson and Ronald Fisher.

pendulum. In these lecture notes the word "model" will generally mean "statistical model", though when we are referring specifically to probabilities (e.g. in the chapter on Randomness, sampling and simulation), we refer to "probabalistic models".

The principle of least squares To tune the parameters, we need to have a measure of how close the line is to the data. One way of defining "close" is how the mathematicians Gauss[3] and Legendre did around 1800: the sum of the squared deviations between the predicted and actual values of y for every value of x. The closeness to the line is a function of β_0 and β_1:

$$f(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 \tag{10.2}$$

We call this function a **error function** or **loss function**. We are searching for the values of β_0 and β_1 that minimise the error function. We denote these values $\hat{\beta}_0$ and $\hat{\beta}_1$, pronounced "beta 0 hat" and "beta 1 hat" respectively. In statistics, the hat denotes the best estimate of a quantity. Minimising this error function is to the principle of least squares, which states that the best fit is obtained by minimising the sum of the squared deviations between the predicted and actual values of y.

Applying the principle of least squares We could try to find the values of $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimise the error function by trial and error, or by using a numerical minimisation routine. However, in the case of the linear regression model, we can find $\hat{\beta}_0$ and $\hat{\beta}_1$ analytically.[4]

We can find values of $\hat{\beta}_0$ and $\hat{\beta}_1$ by setting the partial derivatives[5] of the error function f with respect to β_0 and β_1 equal to 0:

$$\begin{aligned} \frac{\partial f}{\partial \beta_0} &= \sum_{i=1}^{n} (-2)(y_i - \beta_0 - \beta_1 x_i) = 0 \\ \frac{\partial f}{\partial \beta_1} &= \sum_{i=1}^{n} (-2x_i)(y_i - \beta_0 - \beta_1 x_i) = 0 \end{aligned} \tag{10.3}$$

We can rearrange these formulae so that it is obvious that they are a pair of simultaneous equations in β_0 and β_1:

$$\begin{aligned} n\beta_0 + \left(\sum x_i\right)\beta_1 &= \sum y_i \\ \left(\sum x_i\right)\beta_0 + \left(\sum x_i^2\right)\beta_1 &= \sum x_i y_i \end{aligned} \tag{10.4}$$

These equations are called the **normal equations**. Notice that we've used the abbreviated notation for summation here.

The sums over x_i and y_i are related to their sample means: $\sum x_i = n\overline{x}$ and $\sum y_i = n\overline{y}$. These relationships allow us to simplify the normal equations:

$$\begin{aligned} n\beta_0 + n\overline{x}\beta_1 &= n\overline{y} \\ n\overline{x}\beta_0 + \left(\sum x_i^2\right)\beta_1 &= \sum x_i y_i \end{aligned} \tag{10.5}$$

We can then eliminate β_0 to give:

$$\hat{\beta}_1 = \frac{\sum x_i y_i - n\overline{x}\,\overline{y}}{\sum x_i^2 - n\overline{x}^2} = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sum (x_i - \overline{x})^2} \tag{10.6}$$

[3]Gauss was a polymath – and not a eugenicist.

[4]From here until the next section, "Properties of the linear regression line" is not examinable. However, it may be of interest if you want to understand where the expressions for the regression coefficients come from.

[5]Don't worry if you've not met partial derivatives yet. In short, when we find the partial derivative with respect to β_0, we're just imagining that β_0 is the only variable, and β_1, x_i and y_i are all constants. If this section doesn't make sense, skip ahead.

The second part of this equation follows from the identity $\sum_{i=1}^{n}(x-\overline{x})^2 = \sum_{i=1}^{n} x_i^2 - n\overline{x}^2$ (see Equation 3.10 in the Descriptive statistics topic for proof of the identity). It follows from the first of Equation 10.5 that:

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x} \tag{10.7}$$

Properties of the linear regression line Don't worry if you don't follow the derivation above – let's stand back and look at some properties of the least squares linear regression line.

1. The line passes through the point $(\overline{x}, \overline{y})$. We can see this by substituting the value of $\hat{\beta}_0$ into Equation 10.1, which leads to:
$$y = \hat{\beta}_0 + \hat{\beta}_1 x = \overline{y} - \hat{\beta}_1\overline{x} + \hat{\beta}_1 x = \overline{y} + \hat{\beta}_1(x - \overline{x}) \tag{10.8}$$
When $x = \overline{x}$ it follows that $y = \overline{y}$. In Figure 10.3 you can indeed see the regression line passing through the mean of both variables.

2. Equation 10.6 for the gradient of the line $\hat{\beta}_1$ looks similar to the equation for the correlation coefficient r (Equation 6.2 in the Data collection and statistical relationships topic) but is different in one respect. The numerator is the same, but the denominator contains only the sum of squared deviations of x rather than the product of the square roots of the sum of squared deviations of x and y. We can in fact relate $\hat{\beta}_1$ and r via the standard deviations of x and y:
$$\hat{\beta}_1 = \frac{s_y}{s_x} r \tag{10.9}$$

3. If we plug this expression for $\hat{\beta}_1$ into the fitted model (Equation 10.8), we get:
$$y = \overline{y} + \frac{s_y}{s_x} r(x - \overline{x}) = \overline{y} + s_y r\left(\frac{x - \overline{x}}{s_x}\right) \tag{10.10}$$
which rearranges to
$$\frac{y - \overline{y}}{s_y} = r\left(\frac{x - \overline{x}}{s_x}\right) \tag{10.11}$$
Here the fractional terms on the left- and right-hand sides are the standardised versions of the variables x and y, which we also refer to as their z-scores (see section on Correlation). This shows that we can think of making predictions from regression in for steps: (1) compute the z-score for x (i.e. we standardise x) (2) multiply the z-score with the correlation coefficient, (3) multiply-in the scale of y (via s_y), and (4) add-in the mean of y. This is a simple example of moving into a "normalised space" (the z-score), doing the prediction there, and then going to the target space by multiplying-in the y-scale (i.e. the standard deviation) and adding-in the y-location (i.e. the mean).

   This view also shows that "learning", i.e. estimating r, happens in the normalised space, since we correlate the z-score of x with the z-score of y.

Insights from regression with standardised variables We can gain some insights into regression by standardising x and y (Figure 10.5):

1. Since the mean of standardised variables is zero, the regression line passes through (0, 0).
2. The gradient of the regression line of y on x is $\hat{\beta}_1 = r$, from Equation 10.9. The intercept of the regression line $\hat{\beta}_0 = 0$, from Equation 10.7. As the relationship between the variables gets weaker (r gets smaller), the gradient of the regression line decreases.
3. The predicted standardised y is always closer to the mean than the standardised x. In other words, parents who are much taller than average are likely to have children who are taller than average, but not by as much as the parents: the height of these children is likely to be closer to the mean height of children than the tall parents' height was to the mean parental height. The same is true of very short parents and their children. Galton characterised this observation as "regression to mediocrity", or regression to the mean.
4. If we try to predict x from y, i.e. we flip the variables, we find the same gradient $\hat{\beta}_1$. However, by plotting on the same set of axes, we see that when $-1 < r < 1$ the regression lines of y on x (orange) and x on y (green) are different. If there is perfect correlation ($r = 1$ or $r = -1$), the two lines overlap.

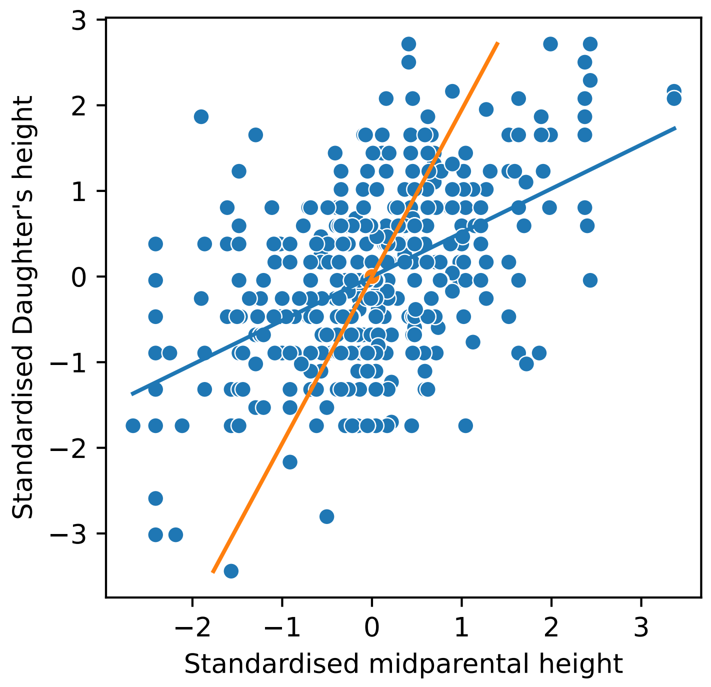


Figure 10.5: Linear regressions of: (i) standardised daughter height on standardised midparental height (orange line) and (ii) standardised midparental height on daughter height (green). For both regression lines $\hat{\beta}_1 = r = 0.51$. Note that children of parents whose height is 3 standard deviations higher than the mean are predicted to be less than 2 standard deviations above the mean – this is what the famous phrase "regression to the mean" is referring to.

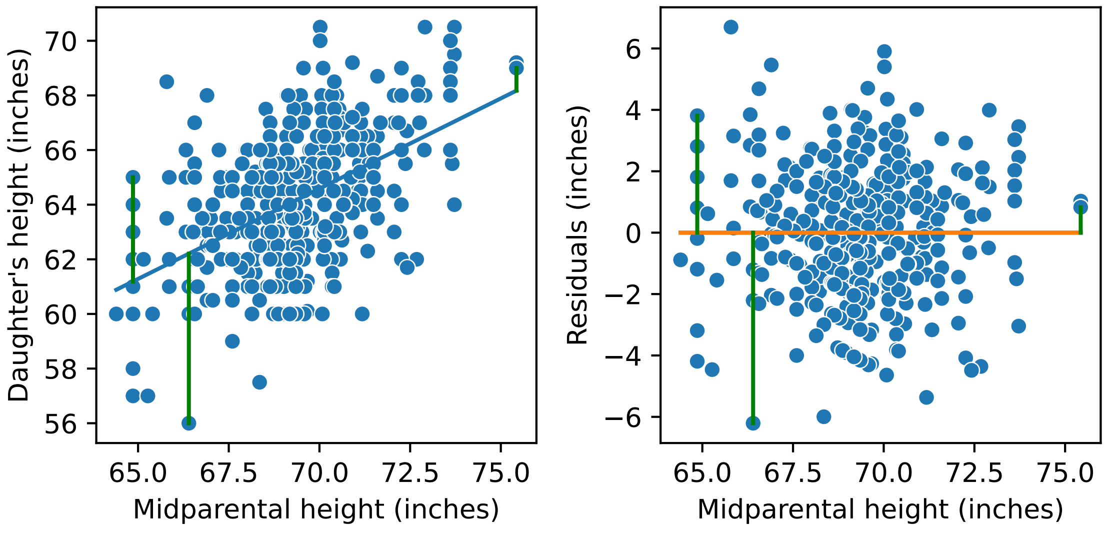


Figure 10.6: Data points and regression line (left) and residual plot (right) for regression of daughter's height on midparental height. The residuals for the same three data points are shown in both plots.

10.3 Visual diagnostics and transformations

Residuals Once we have determined the parameters $\hat{\beta}_0$ and $\hat{\beta}_1$ from the variables x and y, we can compute the **predicted values** $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ of y for each of the x values by substituting x into the estimated regression line:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{10.12}$$

The differences $y_i - \hat{y}_i$ between each pair of predicted and actual values of the response variable are called **residuals**. They can be visualised as the vertical deviations of each data point from the regression line (Figure 10.6, left). Plotting the residuals on their own (Figure 10.6, right) can indicate whether the linear regression model is an appropriate one for the dataset in question.

It is worth noting that residuals from a linear regression always have:

1. Zero mean
2. Zero correlation with predictor variable x
3. Zero correlation with the predicted values $\hat{y}$

These are all true no matter what the data looks like, just like the mean of deviations from the mean is zero.

Nonlinearity The linear regression model is appropriate when the underlying data is linear, but deviations from linearity might not be very apparent when the regression line is plotted with the data. The residual plot makes this more obvious. An example which demonstrates this very clearly is a plot of world population versus year, for the years 1940–2000 (Figure 10.7). The regression line fit looks alright, but when we look at the residuals, we see more clearly that there seems to be something systematically nonlinear going on: the estimate is too low at the start and end of the sequence and the residuals fall and rise again smoothly.

Transforming the data Just because it looks as though data is nonlinear doesn't mean we have to give up on linear regression. We might suppose that the world population grows exponentially (if we looked at a longer time series we would definitely get that idea). To turn an exponential curve back into a linear one, we can transform the data by taking log to the base 10 of the population (Figure 10.8). The residuals now appear to have a less systematic relationship with the predictor variable, and if we turn the residuals of the log population

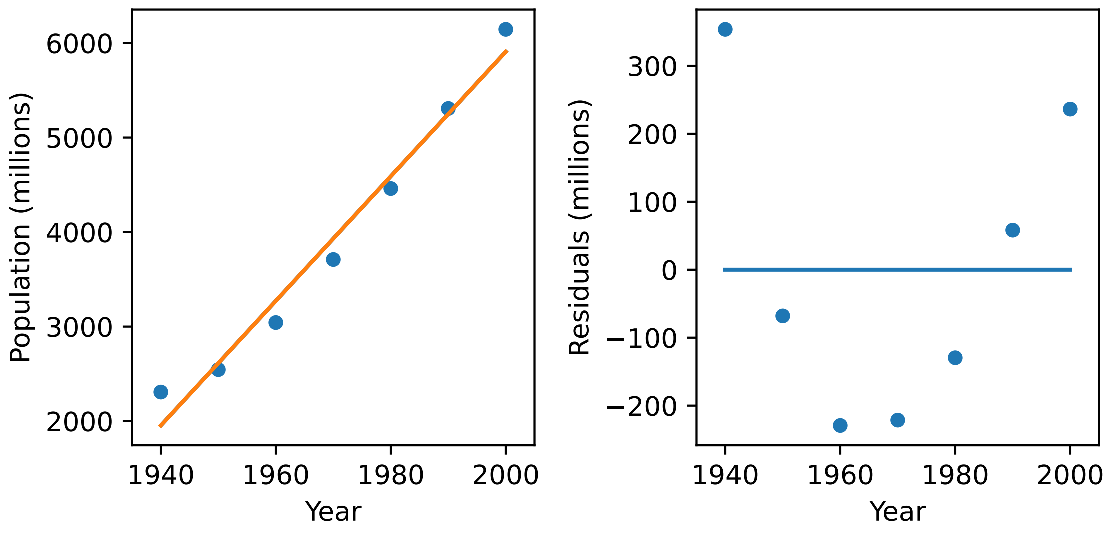


Figure 10.7: World population 1940–2000. Source: HYDE 3.2 database `https://dataportaal.pbl.nl/downloads/HYDE` (Klein Goldewijk et al., 2017)

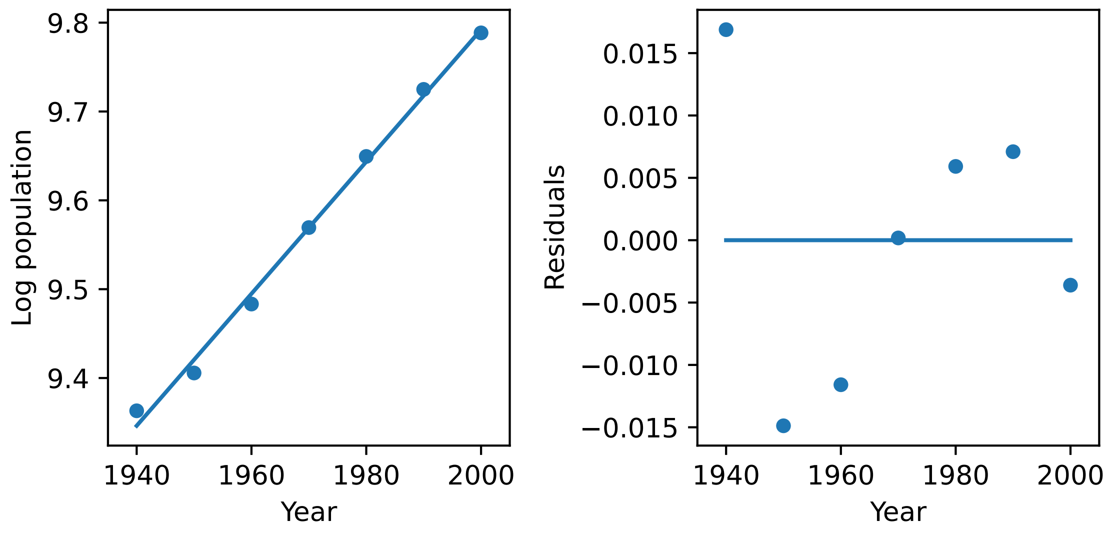


Figure 10.8: Log world population 1940–2000.

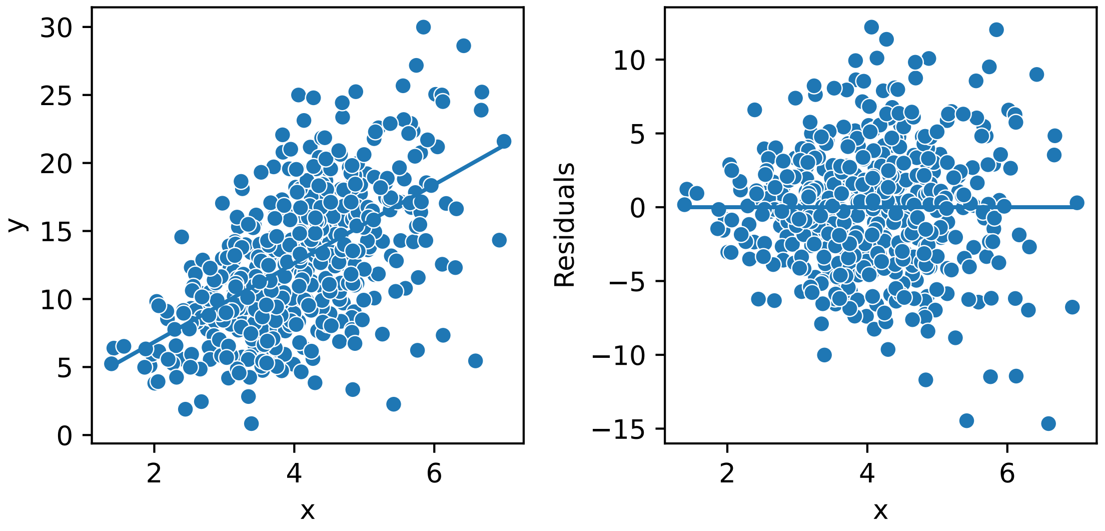


Figure 10.9: A synthetic dataset that exhibits heteroscedasticity.

back into raw numbers of people, we find the largest of these residuals is of the order of 70 million, rather than 300 million for the linear fit (Figure 10.7).

We can also transform the predictor variable or both the predictor and response variables. Whether it is appropriate to transform depends on our understanding of the data and the potential underlying processes.

For example, we might expect that the weight of a squirrel is proportional to the cube of its length (assuming that its height and width are proportional to length). This would suggest regressing the cube root of the weight on the length.

Heteroscedasticity A quick look at the variance of the residuals (Figure 10.6) suggests that the variance of residuals doesn't change as a function of the predictor variable. But suppose we had some data that looks like Figure 10.9. The variance here clearly increases as the predictor variable increases, and we say that the residuals exhibit **heteroscedasticity**. The word heteroscedasticity comes from the ancient Greek "hetero" (different) and "scedastic" (spread) – in other words the variance of the residuals changes as we go along the x-axis.

In the chapter on Linear regression and inference, we'll look at linear regression from a probabilistic perspective, and see that datasets exhibiting heteroscedasticity violate the assumptions that we're using in least-squares regression, namely that the variance of the residuals is independent of x, which we call **homoscedacsticity**. (The Greek word *homos* means "same".)

10.4 Numerical diagnostics

There are a number of ways of measuring how good a regression fit is.

(Root) Mean Square Error We have already seen the sum of the squared deviations; this is what we minimised with respect to the parameters in order to fit the regression line. However, this quantity scales with the number of points, and we would prefer something that gives an indication of the typical size of a residual. The **mean square error** (often written MSE) is defined by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10.13}$$

and the **root mean square error** (RMSE) is just the square root of this. The RMSE gives an indication of how far data points deviate from predictions in absolute terms.

Coefficient of determination Another way of looking at the performance of the regression is to consider how much of the variance in y we can *explain* with x. If we have a perfectly linear relationship (correlation coefficient $r = 1$ or -1), then if we know x, we can predict y precisely. Thus, we have *explained* all the variance s_y^2 of y. When the relationship has a correlation of magnitude lower than 1, if we know x, we can predict y better than if we didn't know x, but we can also expect that the actual value of y will be some way from our prediction, and we say that there is unexplained variance.

The **coefficient of determination** metric quantifies how much variance is explained. It is defined in terms of two quantities:

1. The **total sum of squares** (SST), which we define as the sum of squared deviations from the mean of y:

$$\text{SST} = S_{yy} = \sum (y_i - \overline{y})^2 = (n-1)s_y^2 \tag{10.14}$$

   It is a measure of the total variance in y before we know anything about x.

2. The **sum of squared errors** (SSE):

$$\text{SSE} = \sum (y_i - \hat{y}_i)^2 \tag{10.15}$$

   The SSE is n times the MSE defined in Equation 10.13.

The coefficient of determination is then defined:

$$R^2 = \frac{\text{SST} - \text{SSE}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}} \tag{10.16}$$

R^2 is measure of "goodness of fit". $R^2 = 1$ indicates that the model predicts the data perfectly, whereas a value of 0 indicates no predictive value. In physical and biological sciences we might expect R^2 to be of the order of 0.8, but this can be lower in other disciplines such as social sciences. We can also think of R^2 as 1 minus the MSE normalised by the variance of y. If the mean squared error is a high fraction of the variance, the R^2 will be low.

The notation suggests the coefficient of determination is related to the correlation coefficient r. For a linear regression line we can indeed prove that $R^2 = r^2$. The disadvantage of using R^2 for linear regression is that it doesn't indicate if the correlation is positive or negative. However, R^2 is more versatile: it can measure how well a nonlinear model fits the data. For a nonlinear model, R^2 is not generally equal to the squared correlation coefficient. The R^2 definition also generalises to multiple linear regression, which we will come to in the next section.

Related Python Lab: Linear models

`https://github.com/Inf2-FDS/FDS-S1-06-linear-models`

Chapter 11

Multiple regression

11.1 The principle of multiple regression

Multiple predictor variables Often we want to investigate how a variable depends on more than one predictor variable. For example, we might expect a student's grade in a calculus course could be predicted by their grades in four previous assessments. In this case the response variable is the grade, and we have four predictor variables. The process of predicting a response variable from multiple predictor variables is called **multiple regression**.

Dealing with categorical variables The predictor variables may also be categorical. In the example of predicting a child's height from the heights of their parents, introduced in the chapter on Linear Regression, there were three variables in the data: midparental height, height of child and gender of child. Although gender is a categorical variable, we can convert it to a numeric variable by encoding "Daughter" as 0 and "Son" as 1. As described in the section on Tabular data and variables, this new numeric variable is called a dummy variable or indicator variable, and categorical variables with more than two values (for example colours) can also be encoded using multiple indicator variables. We can treat indicator variables mathematically in the same way that we would treat variables that are naturally numeric.

The multiple regression model To predict the child's height given their midparental height and their gender, we could try to use a non-parametric method, as we did when thinking about regression as prediction. However, we choose here to focus on extending the linear regression model introduced in the last topic, i.e. a parametric model, in which the parameters were the coefficients β_0 and β_1. Suppose we have two predictor variables, $x^{(1)}$ and $x^{(2)}$ and one predictor variable y. (We're using this rather cumbersome notation for the predictor variables so that we don't get confused with the notation for instances of variables.) The multiple regression model is then expressed:

$$y = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} \tag{11.1}$$

Geometrically, this is a 2D plane in 3D space, with β_1 being the gradient of a cut through the plane with constant $x^{(2)}$ and β_2 being the gradient of a cut through the plane with constant $x^{(1)}$.

Principle of fitting the multiple regression model The principle of fitting the multiple regression model is exactly the same as the principle of fitting the linear regression model, just with more variables. We use the least squares principle, but this time we have to adjust three coefficients, β_0, β_1 and β_2, to manoeuvre the plane around so that we minimise the sum of the squared distances between the predicted and actual values of y over all the data points.

In maths, we'll modify the function f (as defined in the Linear Regression topic) that we're minimising to be:

$$f(\beta_0, \beta_1, \beta_2) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}))^2 \tag{11.2}$$

Table 11.1: MSE, RMSE and coefficient of determination for various simple and multiple regression models applied to the heights data.

Model	MSE	RMSE	R^2
Child's height on midparental height and gender code (mult. regression)	4.69	2.16	0.63
Daughter's height on midparental height (regression)	4.08	2.02	0.26
Son's height on midparental height (regression)	5.27	2.30	0.23
Child's height on midparental height (regression)	11.48	3.39	0.10

Note that x_{ij} means the ith instance of the jth variable. We could have used the more cumbersome notation $x_i^{(j)}$, but x_{ij} is simpler, and also makes more sense when we come to minimise the function analytically.

As with linear regression, once we have gone through the maths, we denote the values of the coefficients that minimise f to be $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$, and we can compute the predicted value of the response variable for any values of predictor variables as:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x^{(1)} + \hat{\beta}_2 x^{(2)} \tag{11.3}$$

For now, we will skip the derivation of how to find the regression coefficients – it is sufficient to know that it can be done using an extension to the derivation shown in the Linear Regression slides. If you're interested, we give the derivation in the topic *Mathematics of multiple regression*.

11.2 Interpreting multiple regression coefficients and metrics

Interpretation of the coefficients We'll return to Galton's height data. We'll name the variables as follows:

- $x^{(1)}$: midparental height in inches
- $x^{(2)}$: gender (0 for "Daughter" and 1 for "Son")

After fitting the regression function (Equation 11.1) we find the coefficients:

- $\hat{\beta}_0 = 16.41$ inches: the intercept – nominally the height of a child born to parents with zero height!
- $\hat{\beta}_1 = 0.69$: for every inch of midparental height, we expect the child to be 0.687 inches taller.
- $\hat{\beta}_2 = 5.21$ inches: we expect sons to be 5.21 inches taller than daughters.

(Root) mean squared error Since the MSE and RMSE (see the Linear Regression topic) just depend on y_i and $\hat{y}_i$, we can compute them using the same formulae. We find the MSE and RMSE in Table 11.1, in which we've also included the MSE and RMSE from:

- the regression of daughter's height on midparental height shown in the Linear Regression topic
- the regression of son's height on midparental height (equivalent to the regression of the daughter's height)
- the regression of child's height on midparental height, i.e. as if we did not know the gender of the child.

The RMSE and MSE for the multiple model is mid-way between the values for the two single regression models where we know the gender. This is consistent with the picture of what looks like two regression lines (Figure 11.1). However, if we don't know the gender (imagine we make the prediction before the child is born and without the benefit of a prenatal scan), then the MSE and RMSE are much bigger. This is because the regression line goes midway between the lines that we find for the single-sex regressions or the multiple regressions, so there is more spread around the line.

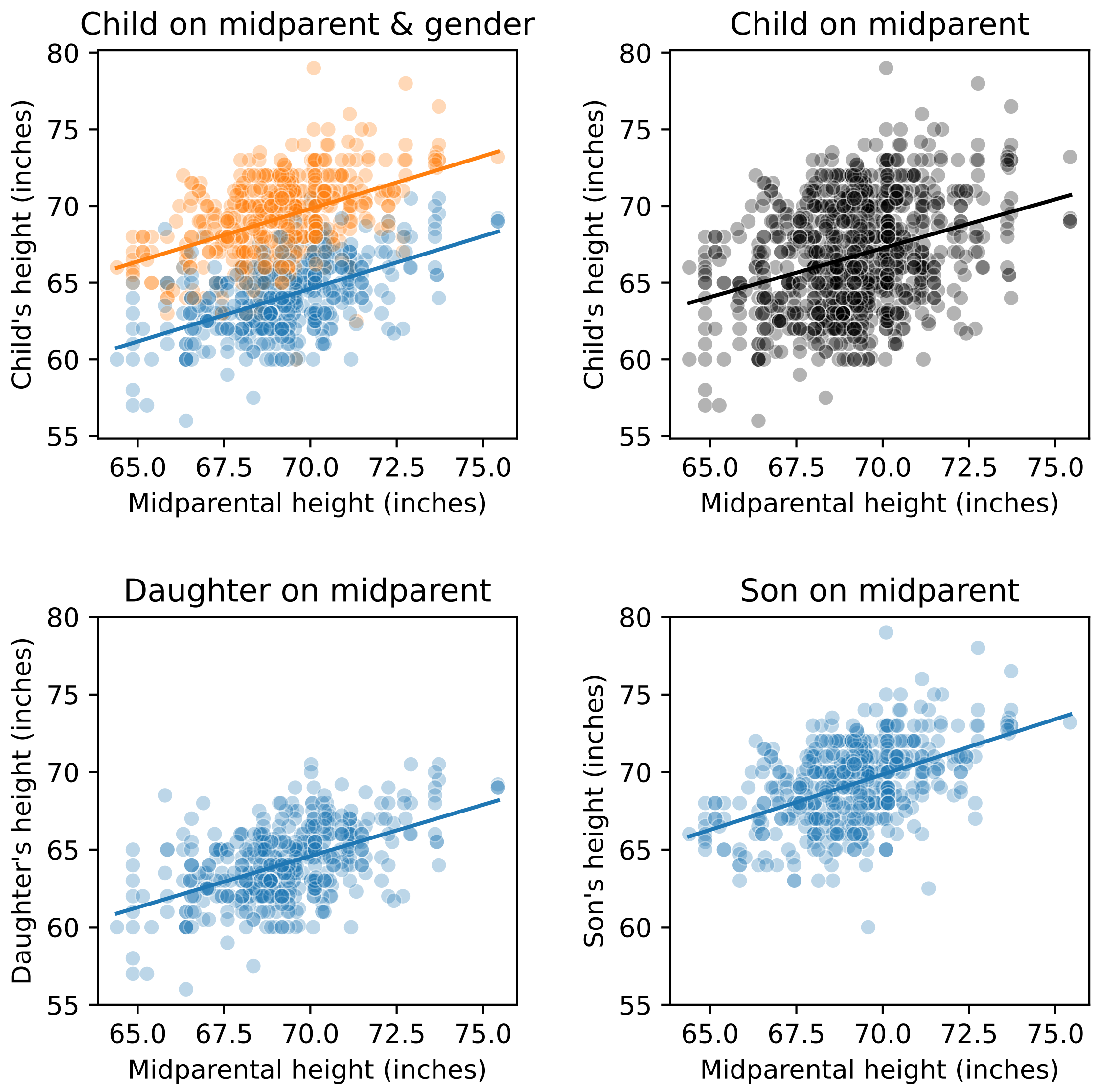


Figure 11.1: Multiple and single regression models of the height data. **Child on midparent & gender:** Multiple regression of child's height on midparent height and child's gender. **Child on midparent:** Single linear regression of child's height on midparent's height. Gender is not a variable, hence the black and grey colour coding. **Daughter on midparent** and **Son on midparent:** Single linear regression of daughter's height on daughter's midparent height (as in last topic) or son's height on son's midparent height.

Interpretation of coefficient of determination Since the coefficient of determination (see the Linear Regression topic) just depends on y_i, $\hat{y}_i$ and $\overline{y}$, we can compute it using the same formula, and here we find that $R^2 = 0.633$. This is a lot more than the $R^2 = 0.263$ we found for regressing daughter's height on midparental height. Does this indicate that the multiple regression model is a better fit? That doesn't seem to make sense, as we've just seen the MSE and RMSE of the multiple regression model is about the same as for the single-sex regression models.

The reason is in that the multiple regression, the mean height that is in the total sum of squares is the mean of *all* children, making the total sum of squares (SST) term in the coefficient of determination bigger. As we have more-or-less the same sum of squared errors (SSE) after fitting, there's a much bigger improvement SST − SSE, and therefore a higher value for R^2.

This isn't the whole picture about the coefficient of determination – we'll return to it later.

Lurking variables Suppose that in the topic on linear regression, we had ignored the gender, and tried to predict the height of sons and daughters just based on their midparental height. In this case we'd have had much higher MSE and RMSE (see table above) and the coefficient of determination would have been 0.10, since there is more variance measured around the mean height of daughters and sons. If we had not considered gender in the simple linear regression, gender would have been a **lurking variable**. In the simple linear regression, we controlled for the lurking variable of gender by only considering the midparental heights of daughters.

Using multiple regression to control for lurking variables By including gender in the multiple regression analysis, we have also controlled for it, and quantified the size of the effect of gender – a double win. However, one side-effect of this model is that the gradients of both regression lines have to be the same, and multiple regression will find a value of $\hat{\beta}_1$ that is not quite optimal for either sons or daughters.

In terms of accurate prediction, we could do better by having two linear regression models: one for the daughters and one for the sons.

11.3 Interaction terms and nonlinear fits

Interaction terms to allow more flexibility There's an alternative to having two models: introduce **interaction terms**, with an extra coefficient, β_3:

$$y = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \beta_3 x^{(1)} x^{(2)} \tag{11.4}$$

In the fourth term, we've effectively introduced a new variable $x^{(3)} = x^{(1)}x^{(2)}$, and we can just feed $x^{(3)}$ into the machinery for computing the coefficients. This new variable allows us to have different gradients for the daughters and sons. It will be zero for daughters (since $x^{(2)} = 0$) but it will be $\beta_3 x^{(1)}$ for sons (since $x^{(2)} = 1$), so the effective height gradient for sons will be $\beta_1 + \beta_3$ rather than just β_1 for daughters.

It turns out that the fits we get are exactly the same as the fit we obtain from two separate regression analyses. Why bother then with this interaction term? By using interaction terms, we can avoid having to create separate datasets for each analysis. Also, had $x^{(2)}$ been a continuous variable, we couldn't create a linear regression model for each of its values.

Nonlinear fits The same idea allows us to use linear regression to fit nonlinear curves. Let's go back to our simple linear regression $y = \beta_0 + \beta_1 x$, and suppose that the residual plot shows a quadratic pattern. We can create a new variable x^2 and treat it as a variable in a multiple regression:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \tag{11.5}$$

Effectively, we've started to think of y being a function of x and x^2.

We don't just have to be confined to polynomial functions: we can try other functions too; the problem is choosing the one that is appropriate for the situation. There is also a problem with **over-fitting**: the more flexibility we add to our function the better we can fit – but is the fit really showing something that's generally true about the population data, or is picking up features that have occurred by chance because of the limited size of the sample?

11.4 Interpreting and refining multiple regressions on many variables

We can extend linear regression to many dimensions, but here we'll just increase the number of predictor variables to four. Let's take a look at an example we mentioned at the start: the prediction of student grades. Figure 11.2 shows the data from 80 students. There look to be correlations between all 4 predictor variables and the grade.

Adjusted R^2 We can use the Python `statmodels` package to run linear regression on the plot. When we do this, we can retrieve a lot of information about the fit (Figure 11.3), but we will just focus on two measures:

- `R-squared` = 0.289: This is the coefficient of determination R^2, as defined in the Linear Regression topic.
- `Adj. R-squared` = 0.251. This is the **adjusted coefficient of determination**, as will be defined below.

If the number of variables is k and the number of instances is n, the adjusted coefficient of determination is given[1]:

$$R^2_{\mathrm{a}} = 1 - \frac{n-1}{n-(k+1)}\frac{\mathrm{SSE}}{\mathrm{SST}} \tag{11.6}$$

This is very similar to the coefficient of determination, but is, by definition, lower than the coefficient of determination – the term $(n-1)/(n-(k+1))$ is bound to be greater than 1, which means that $R^2_{\mathrm{a}} < 1-\mathrm{SSE}/\mathrm{SST} = R^2$.

The reason for adjusting the coefficient of determination is that as we add more and more variables, it becomes easier to fit the data, and therefore the goodness of fit may increase just because of the increase in variables. By adjusting the coefficient of determination, we counteract this tendency.

Meaning of the coefficients Looking at the second table of output in Figure 11.3 we see that:

- Intercept $\hat{\beta}_0 = 36.1215$
- Algebra $\hat{\beta}_1 = 0.9610$
- ACTM $\hat{\beta}_2 = 0.2718$
- ACTNS $\hat{\beta}_3 = 0.2161$
- HSRANK $\hat{\beta}_4 = 0.1353$

The interpretation is that an increase of 1 point in the Algebra test predicts an increase of 0.961 points in the final grade, whereas an increase of 1 in the HSRANK predicts only an increase of 0.135 in the final grade.

Correlated predictor variables The scatter plots (Figure 11.2) show considerable correlation between the predictor variables. We may imagine that if we did a single linear regression with the *Algebra* scores as the predictor variable, we could explain a considerable amount of the variance in the *Grade*. We can check this by re-running the model with just the Algebra as a predictor, in which case we find $R^2_{\mathrm{a}} = 0.231$, not much less than the R^2_{a} for the model with 4 predictor variables. Those extra variables don't seem to have explained much more.

We can also investigate what would happen if we tried to fit using only one of the other variables. When we regress *Grade* on *ACTM*, we find $R^2_{\mathrm{a}} = 0.124$, which is about half the size of R^2_{a} for regressing on *Algebra*. Note that the $R^2_{\mathrm{a}} = 0.247$ when we regress on both *Algebra* and *ACTM* is less than the sum of the two adjusted coefficients of determination ($0.231 + 0.124 = 0.355$). The reason for this is the correlation between *Algebra* and *ACTM*: knowing *Algebra* tells us a lot about *ACTM*, so *ACTM* doesn't add very much new information.

The lesson to learn here is that we need to think carefully about which variables to include as predictor variables in a multiple regression analysis. It may make sense to take out a variable that adds little to the adjusted coefficient of determination. In fact, some of the output in Figure 11.3 that we haven't discussed gives us clues about which variables to drop, but we will discuss that later in the course.

Related Workshop: Interpretation of correlation and linear regression

[1] It would be nice to use the letter D to denote the number of dimensions of the predictor variables, but this is not the convention in the literature on multiple regression. Normally it's either k (as used here and by Devore and Berk (2012)) or p

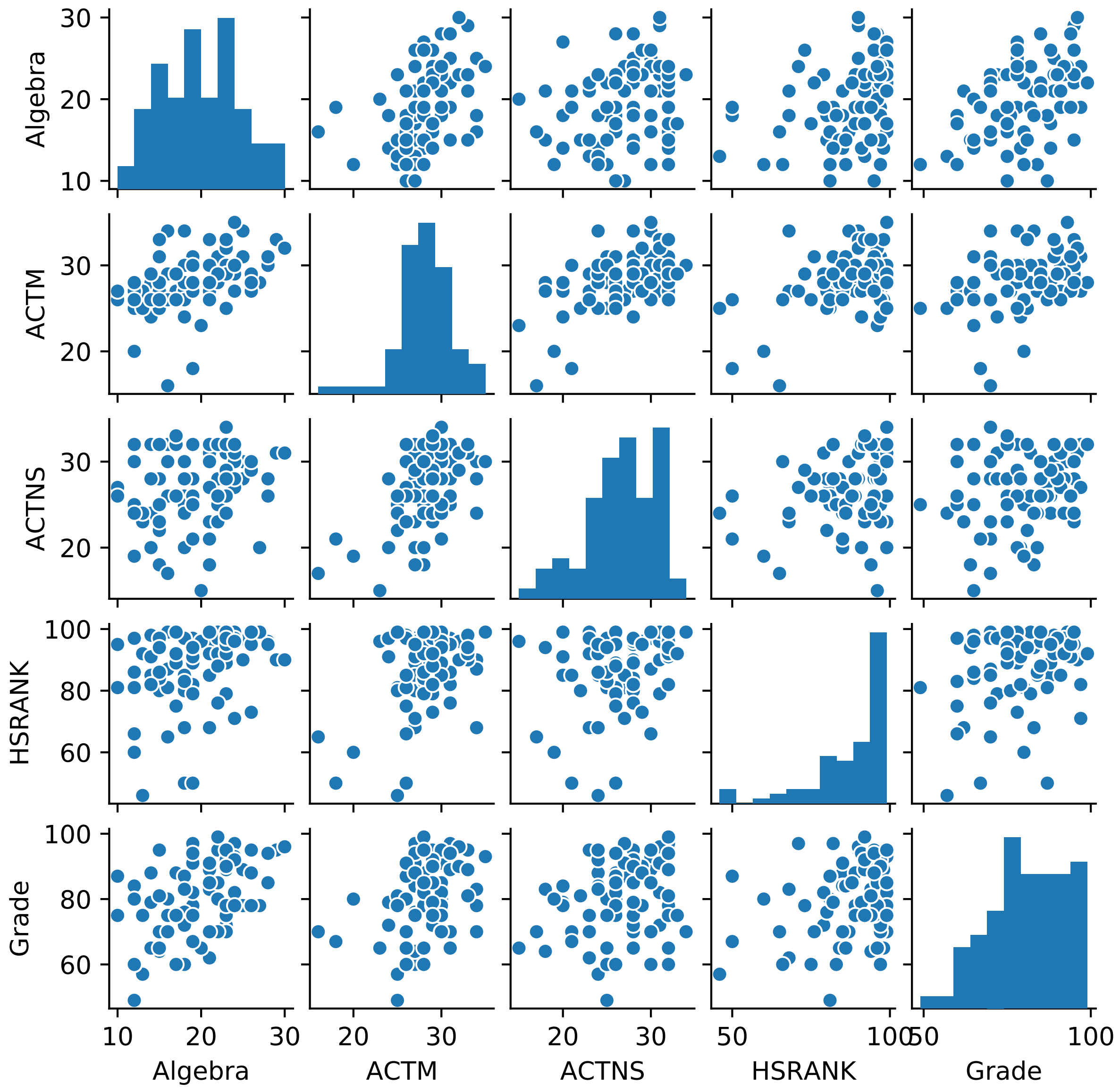
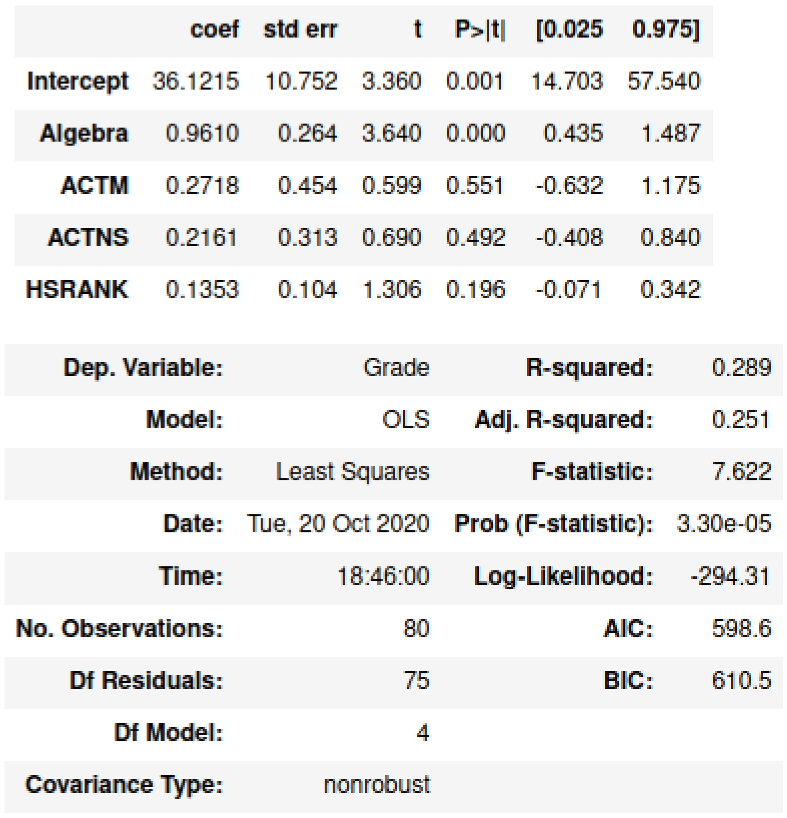


Figure 11.2: Paired scatter plot showing grades of 80 students on a calculus course (**Grade**) and their performance on three earlier tests (**Algebra**, **ACTM** and **ACTNS**) and their high school rank (**HSRANK**). Data from Edge and Friedberg (1984), downloaded from website accompanying Devore and Berk (2012), where it is Example 12.25 `https://extras.springer.com/2012/978-1-4614-0390-6.zip`.



	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	36.1215	10.752	3.360	0.001	14.703	57.540
Algebra	0.9610	0.264	3.640	0.000	0.435	1.487
ACTM	0.2718	0.454	0.599	0.551	-0.632	1.175
ACTNS	0.2161	0.313	0.690	0.492	-0.408	0.840
HSRANK	0.1353	0.104	1.306	0.196	-0.071	0.342

Dep. Variable:	Grade	**R-squared:**	0.289
Model:	OLS	**Adj. R-squared:**	0.251
Method:	Least Squares	**F-statistic:**	7.622
Date:	Tue, 20 Oct 2020	**Prob (F-statistic):**	3.30e-05
Time:	18:46:00	**Log-Likelihood:**	-294.31
No. Observations:	80	**AIC:**	598.6
Df Residuals:	75	**BIC:**	610.5
Df Model:	4		
Covariance Type:	nonrobust		

Figure 11.3: Output from the `statmodels.formula.api.ols` routine applied fitting a multiple regression of **Algebra**, **ACTM**, **ACTNS** and **HSRANK** on **Grade** (Figure 11.2). At this point, we will focus on `R-squared`, `Adj. R-squared` and the `coeff` column.

Chapter 12

Mathematics of multiple regression

12.1 Derivation of coefficients in multiple regression

Derivation of multiple regression coefficients, part 1 We start by repeating Equation 11.1 in the Multiple regression chapter for the regression line with two predictor variables:

$$y = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} \tag{12.1}$$

We could work generally with k variables, but we will stick with 2 variables for now, as we feel it gives a better intuition about what's going on. To find values of $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$ that minimise:

$$f(\beta_0, \beta_1, \beta_2) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}))^2 \tag{12.2}$$

we will modify the method we used for simple linear regression (Linear regression chapter). We could set the partial derivatives of the error function (or loss) f with respect to β_0, β_1 and β_2 equal to 0, and work from the resulting three equations. However, we get some nice insights by proceeding in two steps. First we'll set the partial derivative of f with respect to β_0 to be 0:

$$\frac{\partial f}{\partial \beta_0} = \sum_{i=1}^{n} (-2)(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2}) = 0 \tag{12.3}$$

All the x_{ij} and y_i are constants, so this is an equation in β_0, β_1 and β_2. If we divide through by $-2n$ and define the mean of the jth predictor variable $\overline{x}^{(j)} = 1/n \sum_i x_{ij}$ we get:

$$\begin{aligned} \overline{y} - \beta_0 - \beta_1 \overline{x}^{(1)} - \beta_2 \overline{x}^{(2)} = 0 \\ \Rightarrow \beta_0 = \overline{y} - \beta_1 \overline{x}^{(1)} - \beta_2 \overline{x}^{(2)} \end{aligned} \tag{12.4}$$

We now substitute this expression for β_0 into Equation (12.1) for the regression line:

$$\begin{aligned} y =& \overline{y} - \beta_1 \overline{x}^{(1)} - \beta_2 \overline{x}^{(2)} + \beta_1 x^{(1)} + \beta_2 x^{(2)} \\ y - \overline{y} =& \beta_1 (x^{(1)} - \overline{x}^{(1)}) + \beta_2 (x^{(2)} - \overline{x}^{(2)}) \end{aligned} \tag{12.5}$$

This equation shows that the regression plane will pass through the point $(\overline{x}^{(1)}, \overline{x}^{(2)}, \overline{y})$, which is what happens with the linear regression line in simple linear regression. It is as though the plane is pinned to this location, and there are only two coefficients left to adjust: β_1 and β_2.

It will simplify the following analysis to define new versions of the variables in which the mean has been subtracted:

$$y_i^* = y_i - \overline{y} \quad , \quad x_{ij}^* = x_{ij} - \overline{x}^{(j)} \tag{12.6}$$

Here we've used the stars to denote that this is a version of the variable with the mean subtracted – the star does not mean "complex conjugate"!

Derivation of multiple regression coefficients, part 2 We can now start again from the least squares function f^* that is a function of β_1 and β_2, and which contains our mean-subtracted variables:

$$f^*(\beta_1, \beta_2) = \sum_{i=1}^{n} (y_i^* - (\beta_1 x_{i1}^* + \beta_2 x_{i2}^*))^2 \tag{12.7}$$

We can now partially differentiate with respect to β_1 and β_2 to give the normal equations:

$$\begin{aligned} \frac{\partial f^*}{\partial \beta_1} &= \sum_{i=1}^{n} (-2x_{i1}^*)(y_i^* - \beta_1 x_{i1}^* - \beta_2 x_{i2}^*) = 0 \\ \frac{\partial f^*}{\partial \beta_2} &= \sum_{i=1}^{n} (-2x_{i2}^*)(y_i - \beta_1 x_{i1}^* - \beta_2 x_{i2}^*) = 0 \end{aligned} \tag{12.8}$$

We can divide both sides of the normal equations by -2 and then rewrite them as one matrix equation:

$$\begin{aligned} &\begin{pmatrix} \sum_{i=1}^{n} x_{i1}^*(y_i^* - \beta_1 x_{i1}^* - \beta_2 x_{i2}^*) \\ \sum_{i=1}^{n} x_{i2}^*(y_i^* - \beta_1 x_{i1}^* - \beta_2 x_{i2}^*) \end{pmatrix} = 0 \\ \Rightarrow \quad &\begin{pmatrix} \sum_{i=1}^{n} x_{i1}^*(\beta_1 x_{i1}^* + \beta_2 x_{i2}^*) \\ \sum_{i=1}^{n} x_{i2}^*(\beta_1 x_{i1}^* + \beta_2 x_{i2}^*) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_{i1}^* y_i^* \\ \sum_{i=1}^{n} x_{i2}^* y_i^* \end{pmatrix} \end{aligned} \tag{12.9}$$

Now suppose that we define the matrix $\mathbf{X}$ to be an $n \times 2$ matrix in which the first column contains the n values of x_{i1}^* and the second column contains the n values of x_{i2}^*. We call this matrix the **design matrix** or **regressor matrix** We'll define $\mathbf{y}$ to be the vector containing the n values of y_i^*. By the definition of matrix multiplication, you should be able to verify that the right-hand side of the equation is equal to the matrix product $\mathbf{X}^T\mathbf{y}$:

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} \sum_{i=1}^{n} x_{i1}^* y_i^* \\ \sum_{i=1}^{n} x_{i2}^* y_i^* \end{pmatrix} \tag{12.10}$$

($\mathbf{X}^T\mathbf{y}$ is referred to as the **moment matrix**.)

We can also rewrite the left-hand side in terms of a matrix multiplication of a 2×2 matrix and a vector of coefficients:

$$\begin{pmatrix} \sum_{i=1}^{n} x_{i1}^*(\beta_1 x_{i1}^* + \beta_2 x_{i2}^*) \\ \sum_{i=1}^{n} x_{i2}^*(\beta_1 x_{i1}^* + \beta_2 x_{i2}^*) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_{i1}^* x_{i1}^* & \sum_{i=1}^{n} x_{i1}^* x_{i2}^* \\ \sum_{i=1}^{n} x_{i2}^* x_{i1}^* & \sum_{i=1}^{n} x_{i2}^* x_{i2}^* \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \tag{12.11}$$

There are two things to note about the 2×2 matrix:

1. If you look back to the definitions of sample variance and sample covariance (Equation 6.1 in the chapter on Linear Regression), you will see that its diagonal elements are $(n-1)$ times the variances of $x^{(1)}$ and $x^{(2)}$ and its off-diagonal elements are $(n-1)$ times the covariance of $x^{(1)}$ and $x^{(2)}$.

2. The matrix can be written as $\mathbf{X}^T\mathbf{X}$, which can be referred to as the **normal matrix**.

3. We define the **covariance matrix** to be $\mathbf{S} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$

We can now rewrite Equation 12.9 as a matrix equation:

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y} \tag{12.12}$$

We can then solve it for the vector of coefficients:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{12.13}$$

The vector on the left contains the values of $\hat{\beta}_1$ and $\hat{\beta}_2$, which we can substitute back into Equation 12.4 to get $\hat{\beta}_0$.

Notes on our derivation The derivation can be extended to k predictor variables – Equation 12.13 looks the same, but $\hat{\beta}$ is a $k \times 1$ vector, and the design matrix X is an $n \times k$ matrix. The normal matrix (and covariance matrix) end up being $k \times k$ matrices.

It's worth noting that there are other ways of deriving the coefficients – many treatments in textbooks add a column of 1s to the design matrix and include β_0 in the vector of coefficients. We've chosen not to do this, as it makes the connection with the covariance matrix clearer.

It's also worth noting that although you can program this equation yourself, real-world multiple regression routines use other matrix formulations for reasons of efficiency and numerical stability.

12.2 Interpreting the equation for the coefficients

Equation 12.13 is elegant but also a bit abstract, so we'll try to interpret what the terms in it mean. To do so we'll consider 2 cases, sticking with 2 predictor variables for simplicity.

Interpretation of the derivation 1: no covariance We'll now suppose that the s_1^2, the variance of $x^{(1)}$, and s_2^2, the variance of $x^{(2)}$, are non-zero, but the covariances are 0 (Figure 12.1, left), so the covariance matrix is:

$$\mathsf{S} = \begin{pmatrix} s_1^2 & 0 \\ 0 & s_2^2 \end{pmatrix} = \frac{1}{n-1}\mathsf{X}^T\mathsf{X}$$

We therefore have

$$(\mathsf{X}^T\mathsf{X})^{-1} = \frac{1}{n-1}\mathsf{S}^{-1} = \frac{1}{n-1}\begin{pmatrix} 1/s_1^2 & 0 \\ 0 & 1/s_2^2 \end{pmatrix}$$

and so now our estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$ are

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{(n-1)s_1^2}\sum_{i=1}^n x_{i1}^* y_i^* \\ \frac{1}{(n-1)s_2^2}\sum_{i=1}^n x_{i2}^* y_i^* \end{pmatrix} = \begin{pmatrix} s_{1y}/s_1^2 \\ s_{2y}/s_2^2 \end{pmatrix} \tag{12.14}$$

Here we've used the notation s_{1y} as shorthand for the covariance of $x^{(1)}$ and y.

Now we'll denote the correlation of $x^{(1)}$ and y by r_{1y}, and remember that the correlation coefficient between two variables is defined as covariance divided by the product of the standard deviations, in this case: $r_{1y} = s_{1y}/(s_1 s_y)$. Therefore, we can write the covariance of $x^{(1)}$ and y in terms of the correlation $s_{1y} = r_{1y}s_1s_y$. We have similar definitions for r_{2y}. Substituting in Equation 12.14 gives us:

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \begin{pmatrix} r_{1y}s_y/s_1 \\ r_{2y}s_y/s_2 \end{pmatrix} \tag{12.15}$$

These are exactly the coefficients we found by doing a single linear regression of y on $x^{(1)}$ and $x^{(2)}$ separately. How much a predictor variable that is not correlated with another predictor variable influences our estimate of y depends purely on its correlation with y.

Interpretation of the derivation 2: the general case Now we'll allow the covariances to be non-zero. We simplify calculations by denoting the correlation between $x^{(1)}$ and $x^{(2)}$ as r_{12}, so $r_{12} = s_{12}/s_1s_2$. Thus, the covariance $s_{12} = r_{12}s_1s_2$, and the covariance matrix can be written as:

$$\mathsf{S} = \begin{pmatrix} s_1^2 & r_{12}s_1s_2 \\ r_{12}s_1s_2 & s_2^2 \end{pmatrix} = \frac{1}{n-1}\mathsf{X}^T\mathsf{X}$$

The inverse of the normal matrix is:

$$(\mathsf{X}^T\mathsf{X})^{-1} = \frac{1}{n-1}\mathsf{S}^{-1} = \frac{1}{n-1}\frac{1}{1-r_{12}^2}\begin{pmatrix} 1/s_1^2 & -r_{12}/(s_1s_2) \\ -r_{12}/(s_1s_2) & 1/s_2^2 \end{pmatrix} \tag{12.16}$$

When we multiply by

$$\mathsf{X}^T\mathbf{y} = (n-1)\begin{pmatrix} r_{1y}s_1s_y \\ r_{2y}s_2s_y \end{pmatrix}$$

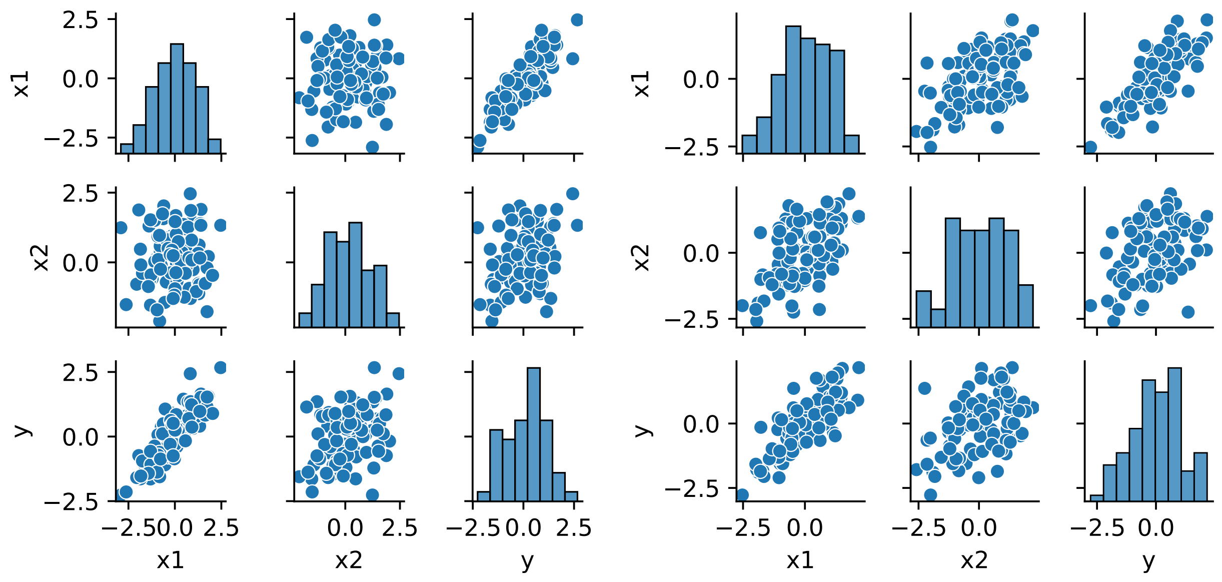


Figure 12.1: Correlated data, generated for the purposes of demonstration. In both plots the correlations between the predictor variables and the response variables are $r_{1y} = 0.8$ and $r_{2y} = 0.4$, and all variables have unit variance. In the left-hand plot the predictor variables are uncorrelated $r_{12} = 0$. The expected regression coefficients are $\hat{\beta}_1 = r_{1y} = 0.8$ and $\hat{\beta}_2 = r_{2y} = 0.4$. In the right-hand plot, the predictor variables have a correlation $r = 0.5$, but due to the correlation between the predictor variables, the regression coefficients change to $\hat{\beta}_1 = 0.8$ and $\hat{\beta}_2 = 0$. See text for details.

we end up with

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \frac{1}{1-r_{12}^2} \begin{pmatrix} r_{1y}s_y/s_1 - r_{12}r_{2y}s_y/s_1 \\ r_{2y}s_y/s_2 - r_{12}r_{1y}s_y/s_2 \end{pmatrix} = \frac{1}{1-r_{12}^2} \begin{pmatrix} (r_{1y} - r_{12}r_{2y})s_y/s_1 \\ (r_{2y} - r_{12}r_{1y})s_y/s_2 \end{pmatrix} \tag{12.17}$$

The coefficient values found in the no-covariance case (Equation 12.15) are still there (when $r_{12} = 0$), but we see that when there is a non-zero correlation between the predictor variables, the coefficients are altered. The no-covariance estimate for $\hat{\beta}_1$ (i.e. r_{1y}) is adjusted by subtracting a fraction r_{12} of the correlation of the response variable with the other predictor variable, r_{2y}. This makes sense, since if didn't make this correction, the contribution $\hat{\beta}_2 x^{(2)}$ would "pollute" our estimate of y. Likewise, $\hat{\beta}_2$ is adjusted by subtracting $r_{12}r_{1y}$.

For example, suppose we have unit variance variables ($s_1 = s_2 = 1$) and $r_{12} = 0.5$, $r_{1y} = 0.8$, $r_{2y} = 0.4$ (Figure 12.1, right). Then we have

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \frac{1}{1-0.5^2} \begin{pmatrix} 0.8 - 0.5 \times 0.4 \\ 0.4 - 0.5 \times 0.8 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0 \end{pmatrix}$$

Essentially the predictor variable with the stronger correlation with the response variable reduces the influence of the predictor variable with the weaker correlation. The more informative predictor variable "wins" the competition to explain the response variable.

There is a special case when r_{12} is non-zero, but $r_{1y} = r_{2y}$. The coefficients become:

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \frac{1-r_{12}}{1-r_{12}^2} \begin{pmatrix} r_{1y}s_y/s_1 \\ r_{2y}s_y/s_2 \end{pmatrix} = \frac{1}{1+r_{12}} \begin{pmatrix} r_{1y}s_y/s_1 \\ r_{2y}s_y/s_2 \end{pmatrix}$$

Here $x^{(1)}$ and $x^{(2)}$ have equal correlation with y, so are equally informative about y. As the correlation between the predictor variables gets stronger the coefficients are scaled down; when r_{11} approaches 1, the coefficients are half what they would be in the case of uncorrelated predictor variables. If two people are singing the same song, you can halve the volume of both singers, and still hear the same information.

Colinearity The extreme case of correlation is $r_{12} = 1$, when $x^{(1)} = cx^{(2)}$, with c being a constant. In this case the denominator of Equation 12.17 is zero. This reflects the fact that the determinant of the covariance matrix $\mathbf{S}$ is zero – the two rows of the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ in Equation 12.11 are multiples of each other. There is therefore no solution to Equation 12.13, and a linear regression function in a stats package like `statsmodels` in Python will complain about a singular matrix.

When r_{12} is large there are still problems, since small differences in the correlation of $x^{(1)}$ and $x^{(2)}$ with y can lead to very different estimates of the coefficients. The interpretation of the coefficients therefore needs particular care.

Chapter 13

Dealing with high dimensions – PCA

Recommended reading

Witten, Frank, Hall and Pal *Data mining*, 4th 3d, pp 304–307 contains an overview of PCA. Different sources use different notation, so it may be least confusing just to follow these notes.

13.1 The principle of Principal Components Analysis (PCA)

The challenges of high dimensions In the multiple regression topic, in the student grade prediction example, we were beginning to see two challenges of dealing with more than one predictor variable:

The challenge of visualisation We can see a lot in the paired correlation plots. With 4 predictor variables, the visualisation works, but what about if we had 26 variables? The Scottish Index of Multiple Deprivation (SIMD, Table 13.1) records 26 variables for each of 6527 data zones in Scotland. A 26×26 grid of scatter plots is going to be difficult to read.

The challenge of interpretation In the grades example, the test grades (predictor variables) were correlated, which made the interpretation of the regression coefficients challenging – and this was with only 4 predictor variables. In the SIMD example, we might expect many of the 26 variables to be correlated, e.g. the time it takes to drive to the nearest primary school and the time it takes to drive to the nearest secondary school.

There is also another problem with high-dimensional data, called the **curse of dimensionality**: essentially a large number of dimensions makes is harder for distance-based methods such as clustering and nearest neighbours to work effectively – we'll come back to the curse of dimensionality in the following lectures on clustering and nearest-neighbour methods.

Table 13.1: Scottish Index of Multiple Deprivation, 2016 edition (Scottish Government, 2016). `https://simd.scot`. It has $n = 6527$ data points (postcode zones), each associated with $D = 26$ variables.

Location	Employ-ment	Illness	Attain-ment	Drive Primary	Drive Secondary	Crime	...
Macduff	10	95	5.3	1.5	6.6	249	...
Kemnay	3	40	5.3	2.4	2.4	168	...
Hilton	0	10	6.3	2.2	3.0	144	...
Ruchill	8	130	4.9	1.7	5.6	318	...
Belmont	2	50	6.1	3.1	3.2	129	...
...	...	...	...	...	...	...	...

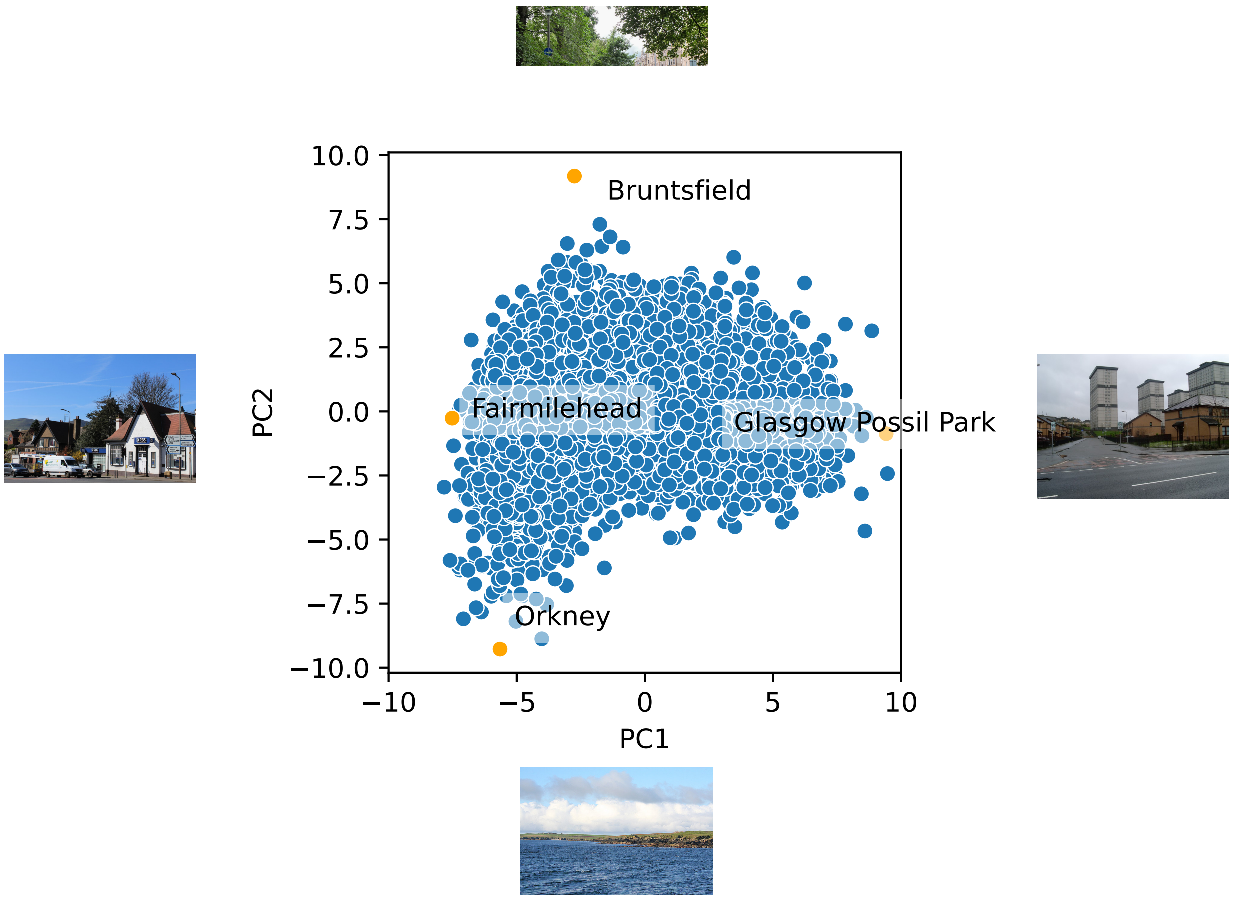


Figure 13.1: Scatter plot of first and second principal component scores (PC1 and PC2) of 6527 data points in the SIMD dataset (blue dots). Locations of 4 data zones are indicated in orange dots next to an image from that data zone. All photos released under CC licence from geograph.co.uk. Credits: Orkney © Des Colhoun; Possil Park © Stephen Sweeney; Brunstfield © Leslie Barrie; Fairmilehead © Jim Barton.

In **dimensionality reduction** methods these challenges are addressed by reducing the number of dimensions in the data while retaining as much useful information as possible. There are a number of dimensionality reduction methods which differ in what aspects of the data they preserve.

Principal components analysis We are going to discuss one method of dimensionality reduction called **principal components analysis** (PCA).

PCA can be applied to a set of D numeric variables with n datapoints. In contrast to linear regression, all variables are treated equally: there is no response variable that we are trying to predict, just a set of variables whose structure we're trying to understand better. The result of PCA is a set of up to D new variables (with n datapoints). We can keep $k \leq D$ of the most informative new variables.

In PCA the objectives are:

1. change the angle we view the data from to see things clearly
2. ignore small details in the data that don't affect the big picture.

We'll specify these objectives more precisely and explain how PCA works later. First, we will show the results when PCA is applied to the SIMD example (Table 13.1).

Example of PCA We can use PCA to reduce the number of variables D in the SIMD data from $D = 26$ to $k = 2$, allowing us to visualise all $n = 6527$ data points (Figure 13.1). In this plot, the ith datapoint has coordinates

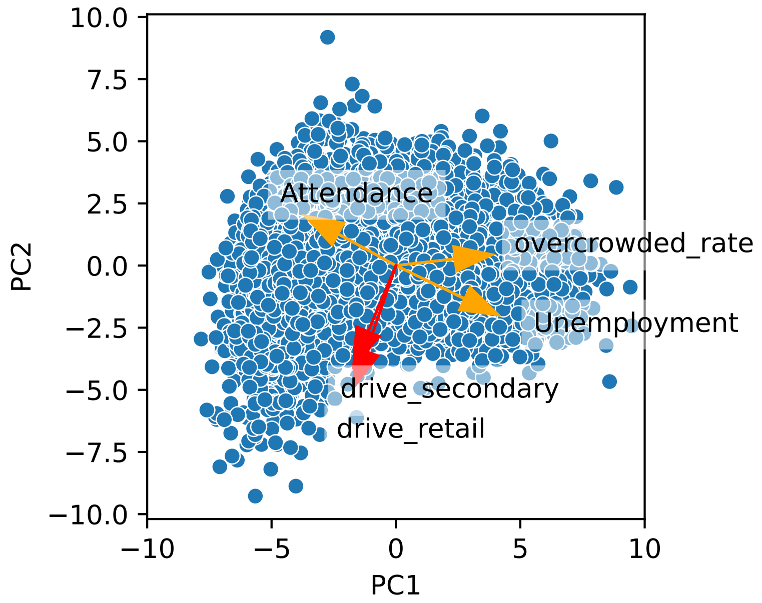


Figure 13.2: Scatter plots of first and second principal component scores of SIMD data zones (blue dots). The projection of three original variables related to deprivation are shown as orange arrows emanating from the origin. High unemployment and overcrowded rate are found in areas with higher deprivation, whereas high school attendance is found in areas with low deprivation. These vectors are more closely aligned with the first principal component (PC1), which we therefore interpret as "Deprivation". Red arrows indicate the projections of the time take to drive to the nearest secondary school or retail outlet. As these are aligned with PC2, we therefore interpret PC2 as being related to distance to services, or "Remoteness".

(t_{i1}, t_{i2}) in which each coordinate is a linear combination of the standardised[1] data z_{ij} shown in Table 13.1:

$$\begin{aligned} t_{i1} &= p_{11}z_{i1} + p_{21}z_{i2} + \cdots + p_{D1}z_{iD} \\ t_{i2} &= p_{12}z_{i1} + p_{22}z_{i2} + \cdots + p_{D2}z_{iD} \end{aligned} \tag{13.1}$$

The weights $p_{11}, p_{21}, \ldots, p_{D1}$ are elements of the **first principal component** and t_{i1} is the first principal **component score** of the ith datapoint; we will explain how to find them later. Likewise, $p_{12}, p_{22}, \ldots, p_{D2}$ form the second principal component and t_{i1} is the second principal **component score** of datapoint i. The weights in the principal component indicate how much influence each original variable has over each principal component score – sometimes they are referred to as **loadings** or **weights**. The axes in Figure 13.1 are labelled PC1 (first principal component – "PC" stands for principal component) and PC2 (second principal component).

To see the influence of each original variable on PC1 and PC2 scores, we can project the jth original variable onto the plot by setting z_{ij} to 1 and all the other z's to 0 in Equation 13.1. In this case, the coordinates we'll be plotting are (p_{j1}, p_{j2}). The orange arrows in Figure 13.2 show the projections of the variables for unemployment, overcrowding (in housing) and school attendance. Unemployment and overcrowding have high PC1 scores. In contrast, school attendance has a low PC1 score. This all makes sense if we identify the first component score with "Deprivation". We can rephrase the previous sentences as "unemployment and overcrowding are found in areas of high deprivation and high school attendance is found in areas of low deprivation".

The red arrows in Figure 13.2 show the projections of the time to drive to the nearest retail outlet and time to drive to the nearest secondary school. These vectors have higher magnitude PC2 scores than PC1 scores. We therefore identify PC2 as being to do with "remoteness" – low values of PC2 indicate the zone is more remote.

[1]Remember from the section on Correlation that we standardise the jth variable $x^{(j)}$ by subtracting its mean $\bar{x}^{(j)}$ and dividing by its standard deviation s_j, so that $z_{ij} = (x_{ij} - \bar{x}^{(j)})/s_j$. Generally, the data we supply to PCA do not need to be standardised, but we still do need to subtract the mean in order to compute the component scores.

Note that the correlation between the PC1 and PC2 scores is zero. It is a general property of PCA there are no correlations between the scores of different principal components.

In this particular example, the visualisation shows a unimodal distribution of data with little obvious structure. Later on in the course we will see examples where PCA reveals clusters of data – though still with zero correlation.

Even if no structure is apparent, reducing the dimensionality of the data can be useful for further analysis. For example, suppose we have data on cancer screening rates in each data SIMD zone, we could then do multiple regression of the cancer screening rate on the new deprivation and remoteness variables. This is probably going to give us coefficients that are a lot more interpretable than regressing on all 26 variables.

Projecting principal component scores back into the data space Suppose we have identified the first two principal component scores t_{i1} and t_{i2} of area i. We might wish to project them back into the data space, to see what the original variables looked like. To do this we can use the following equations to give approximations (indicated by the tilde) to the original standardised variables:

$$\begin{aligned}
\tilde{z}_{i1} &= p_{11}t_{i1} + p_{12}t_{i2} \\
\tilde{z}_{i2} &= p_{21}t_{i1} + p_{22}t_{i2} \\
&\vdots \\
\tilde{z}_{iD} &= p_{D1}t_{i1} + p_{D2}t_{i2}
\end{aligned} \tag{13.2}$$

We can include more terms for higher PCs, right up to the Dth PC. In general, the jth component of the i data point is given:

$$\begin{aligned}
z_{ij} &= p_{j1}t_{i1} + p_{j2}t_{i2} + \dots p_{jD}t_{iD} \\
&= \sum_{k=1}^{D} p_{jk}t_{ik}
\end{aligned} \tag{13.3}$$

Once we've got the standardised variables, we can convert back to the original variables using the formula $x_{ij} = z_{ij}s_j + \overline{x}_j$.

Principal component equations in vector notation The equations used so far may make more sense when expressed as vectors. The jth principal component is actually a vector in the original data space:

$$\boldsymbol{p}_j = (p_{1j}, p_{2j}, \dots, p_{Dj})^T \tag{13.4}$$

All the principal component vectors are orthogonal to each other. With this notation we can write Equation 13.2 as a linear combination of the principal component vectors, weighted by the principal component scores:

$$\boldsymbol{z}_i = t_{i1}\boldsymbol{p}_1 + t_{i2}\boldsymbol{p}_2 + \dots \tag{13.5}$$

The dots indicate that we could go up to $t_{iD}\boldsymbol{p}_D$.

We can rewrite Equation 13.1, in which we computed the scores, as the scalar product of the ith standardised data point and the jth principal component:

$$t_{ij} = \boldsymbol{z}_i \cdot \boldsymbol{p}_j \tag{13.6}$$

We'll extend this notation to matrix notation in the derivation.

13.2 Principle of finding principal components

A 2D example We'll now discuss the principle of how to determine the principal components with an imaginary 2D example. Suppose we ask if is there are different types of Informatics students, perhaps based on their preferences for programming languages and for drinks. We ask students if they prefer, on a scale of 1–9, Haskell (1) to Java (9), and if they prefer Tea (1) to Coffee (9), and find the data in Table 13.2.

Table 13.2: Imaginary data about Informatics students' preferences for programming languages and drinks.

Student ID	Language	Drink
1	9	8
2	3	1
3	8	7
4	2	2
5	3	3
6	8	6
7	2	3
8	8	8
9	1	2
10	6	7

Plotting the data (Figure 13.3 left) shows that students' preferences for drinks and programming languages are correlated. It seems that we could characterise every Informatics student by one number that is low if they like Haskell and tea, and high if they like Java and coffee. If we could rotate the axes (Figure 13.3 right), the new x-axis would give us this number.

Once we've done the rotation (changed the angle), we end up with the data plotted against a new set of axes, which are the principal components (Figure 13.4, top). The new x-axis, which tells us a lot about the students' preferences for Java and coffee or tea and Haskell, is the first principal component (PC1). The new y axis is the second principal component (PC2). It is worth noting two points:

- The correlation between the new PC1 and PC2 scores is zero. It is a general property of PCA that correlations between scores is zero.
- We have not lost any information about the data; we can reconstruct the original data by reversing the rotation. It is a general property of PCA that it is possible to reconstruct the data if scores of all D principal components are retained.

The second principal component doesn't seem so informative, so we could just ignore it altogether (Figure 13.4, bottom). Thus, we have ignored small details in the data that don't affect the big picture. We have performed dimensionality reduction by reducing the number of values describing each data point from two to one.

Objective of rotation There are two questions that we haven't answered so far:

1. How do we choose how much to rotate the axes?
2. What counts as "informative"?

The answer to both questions is "variance". In Figure 13.4 (top), the variance of the data in the PC1 direction is much greater than the variance of the data in the PC2 direction. The high variance PC1 is telling us a lot about the informatics students, whereas the low variance PC2 tells us little. Therefore, in order to choose how to rotate the axes, we use the variance as an objective. In fact there are two ways of formulating PCA:

1. Maximum variance formulation: find an axis that maximises the variance of the data projected onto it
2. Minimum variance formulation: find an axis that minimises the variance of the data projected onto it

It doesn't matter which formulation we use; the answer is the same either way.

Explained variance The variance in the original x (Programming language) and y (Drink) directions was 9.7 and 7.7. The sum of these two variances is the total variance, i.e. 17.4. It turns out that the sum of the variance along the principal components is exactly the same. However, the variance of the PC1 scores is 16.5, i.e. 96% of the total variance. We therefore say the PC1 explains 96% of the variance.

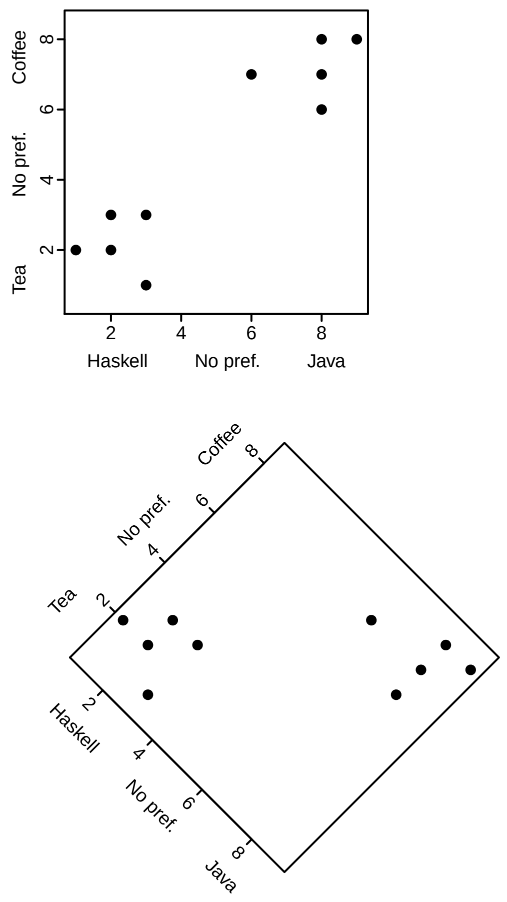


Figure 13.3: Informatics students' preferences for drinks and programming languages, as plotted initially (left), and rotated (right).

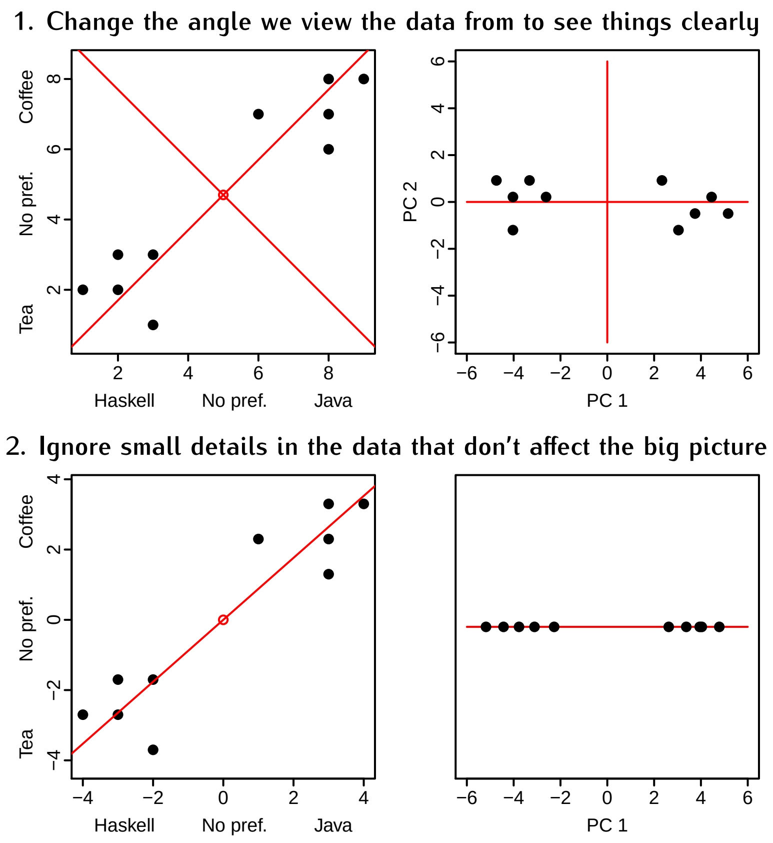


Figure 13.4: Visualisation of how PCA achieves the two objectives in the text.

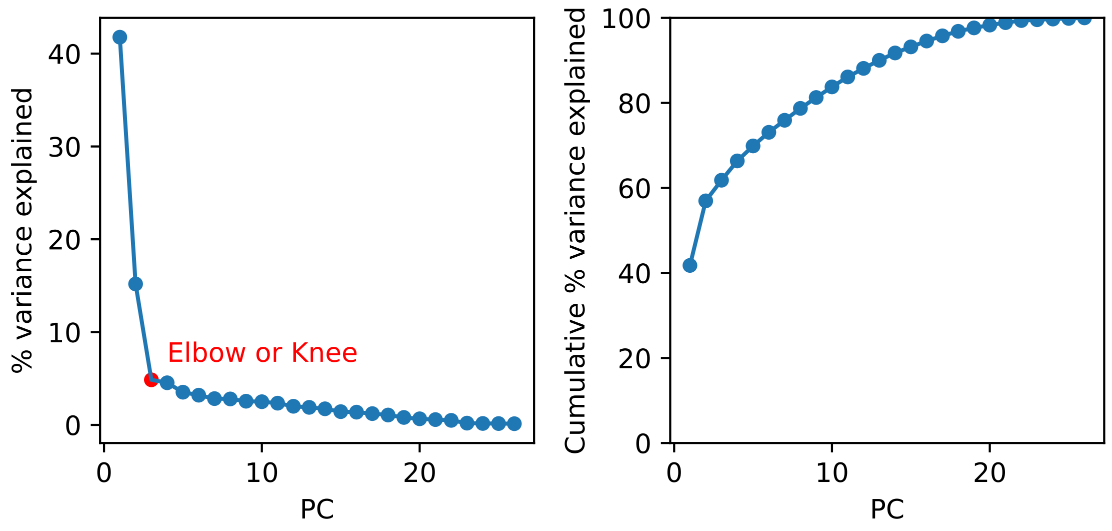


Figure 13.5: Scree plot for PCA applied to SIMD example (left). The elbow (or knee) is indicated in red. The Cumulative scree plot (right).

More than 2D In general, we can find D principal components in D dimensions. The principal components are all orthogonal to each other, and each principal component explains a certain fraction of the variance. We order the principal components from the one that explains most variance to the one that explains least.

In the SIMD example, the first principal component explains 41.7% of the data and the second explains a further 15.2%. Thus, the first two principal components together explain 56.9% of the variance. We can visualise how much each principal component explains in a **scree plot** or **cumulative scree plot** (Figure 13.5).

How many components to choose? Obviously if we are visualising data, we can only look straightforwardly at up to 3 dimensions. The scree plot helps us to choose how many components to include if we are using PCA as a preprocessing step. A rule of thumb is to use the components to the left of the "knee" or "elbow" of the scree plot, i.e. the point where the gradient changes sharply. In Figure 13.5 this point is indicated in red, and the rule of thumb would suggest that we use PC1 and PC2. There are more principled ways of choosing, which we won't cover at this point, and it may also be that successful application of PCA requires more components.

In the next section, we'll look at the maths of how to find the directions of the principal components and the associated variances. However, you should already know enough to skip to the section after, which is about applying PCA to help with a regression problem.

13.3 PCA and regression

PCA as preprocessing PCA is often used as a preprocessing step before another method, e.g. linear regression or K-means. Here we'll see how it can help simplify the grades example from the linear regression lecture. Figure 13.6 shows the results of applying PCA to the predictor variables in this example. Note the correlations between the PC scores are all zero; the general property of PCA already mentioned. However, the correlations between the PC scores and the Grade are non-zero.

We can regress the Grade y on the principal component scores $t^{(1)}, t^{(2)} \dots$:

$$y = \hat{\beta}_0 + \hat{\beta}_1 t^{(1)} + \hat{\beta}_2 t^{(2)} + \dots \tag{13.7}$$

When we regress on all 4 PC scores, we get exactly the same predictions and coefficient of determination as we do for regressing on all variables (Table 13.3). This makes sense, since by keeping all 4 components we have not lost any information about the data. It is more surprising that the coefficient of determination with if we

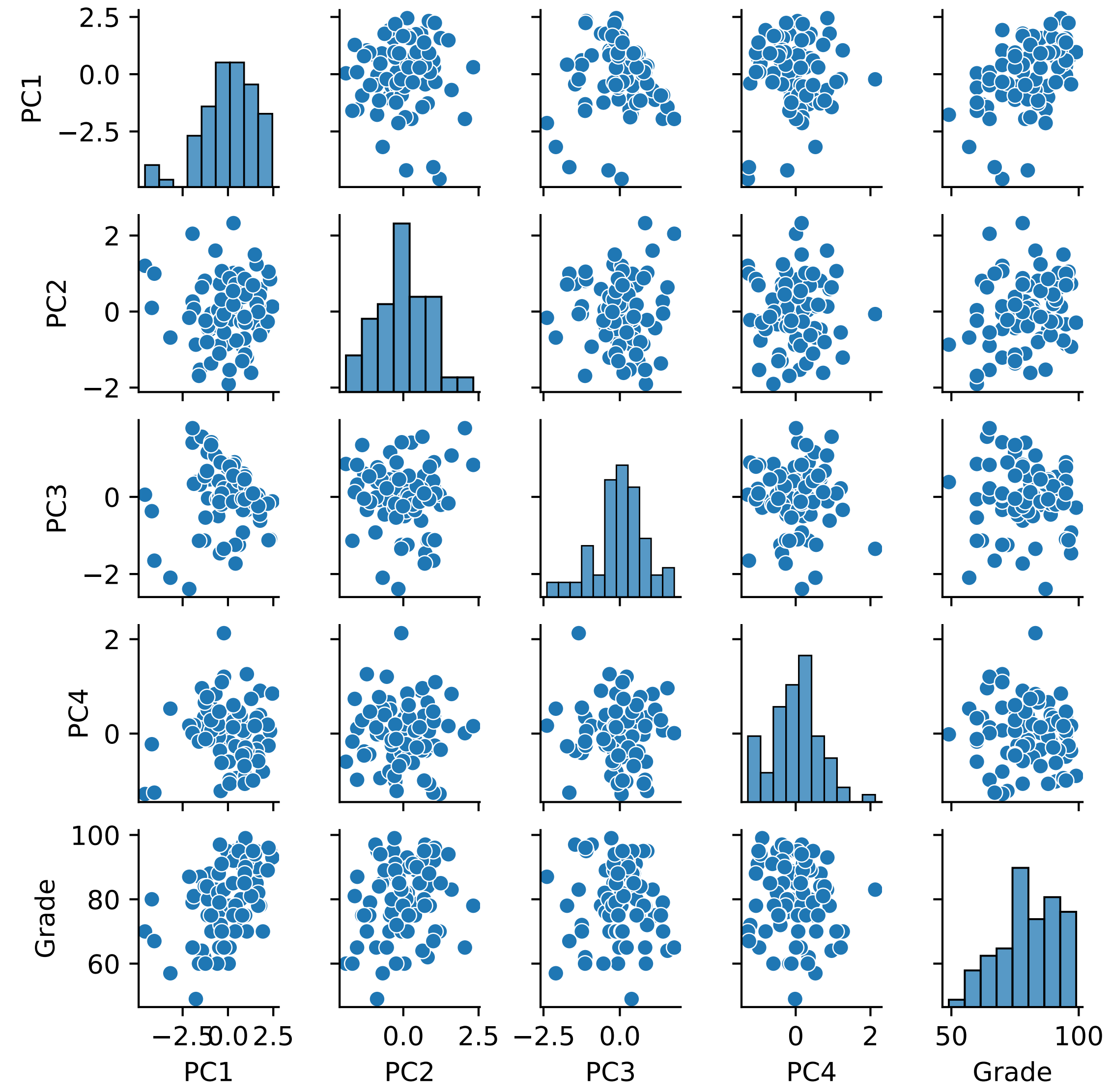


Figure 13.6: PCA scores of predictor variables in Grades example (see Multiple regression lecture notes).

Table 13.3: Coefficient of determination and adjusted coefficient of determination for regression of grades on original variables and on 2 or 4 PC scores.

	4 Original variables	4 PC scores	2 PC scores
R^2	0.289	0.289	0.282
R^2_a	0.251	0.251	0.263

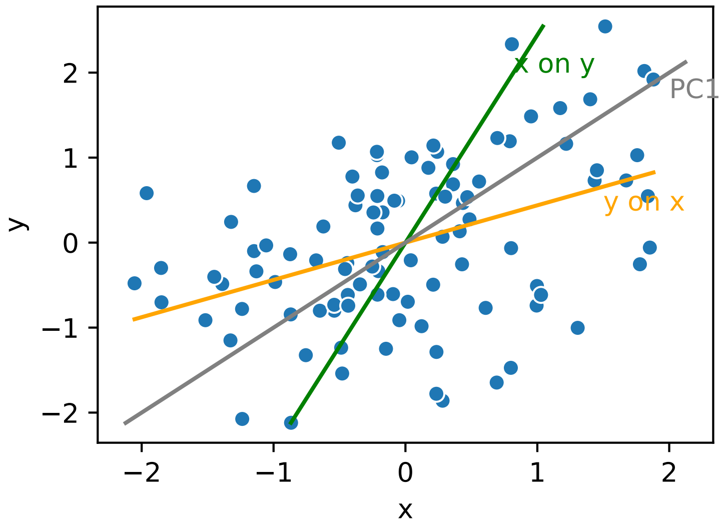


Figure 13.7: Regression of y on x, x on y and the first principal component of data with a correlation coefficient $r = 0.5$.

regress on only the first two PC scores is almost as high. Furthermore, the adjusted coefficient of determination is actually higher when regression on the first two principal components, due to there being fewer variables. There is no combination of any two of the original variables that gives as high a coefficient of determination.

This example demonstrates that PCA can be a useful preprocessing step for regression, by decorrelating the variables.

PCA and linear regression lines Thinking back to linear regression, we remember the distinction between the regression lines of y on x and x on y. In two dimensions there is now a 3rd line: the first principal component. This goes right between the regression lines, and is probably what you would think the line of best fit to the data is. In fact, it is a line of best fit. It's the line that minimises the sum of the squared distances from the data points to the line, rather than minimising the error in predicting y or x.

13.4 Derivation of PCA

Overview of derivation Here are the steps we'll take in our derivation:

1. Define variance along the original axes
2. Project data onto rotated axes
3. Compute variance in these axes
4. Find direction of the axis that maximises variance of data projected onto it (1st principal component, PC 1)
5. Interpret

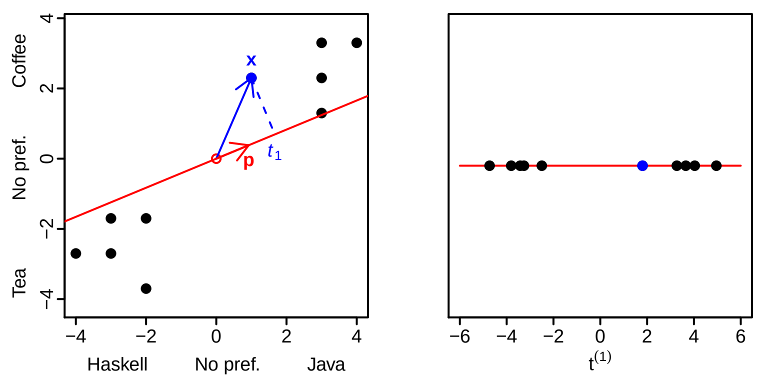


Figure 13.8: Projection of a data point $\boldsymbol{x}$ onto a unit vector $\boldsymbol{p}$.

6. Find the 2nd principal component (PC 2)
7. Quantify what is lost by dimensionality reduction

This list may seem overwhelming, but it actually boils down to about 4 lines of code (assuming some helper functions), shown in Listing 13.1.

Step 1: Defining variance along original axes We've already met a lot of the mathematical machinery we need in the multiple regression topic. We'll assume now that we have D variables $x^{(1)}, \dots, x^{(D)}$, and that we have defined zero-mean versions of the variables $x^*_{ij} = x_{ij} - \overline{x}^{(j)}$. Usually, as in the SIMD example, we start off with standardised variables anyway. We can arrange these zero-mean variables in an $n \times D$ matrix,

$$\mathsf{X} = \begin{pmatrix} x^*_{11} & \cdots & x^*_{1D} \\ \vdots & & \vdots \\ x^*_{n1} & \cdots & x^*_{nD} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^{(1)} & \cdots & \boldsymbol{x}^{(D)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{pmatrix} \tag{13.8}$$

which it can be helpful to write in terms of the D $n \times 1$ vectors representing all the data in each dimension, or as the transposes of the n $D \times 1$ vectors representing each data point.

We've also met the covariance matrix, the $D \times D$ matrix that's derived from the data matrix:

$$\mathsf{S} = \begin{pmatrix} s_{11} & \cdots & s_{1D} \\ \vdots & & \vdots \\ s_{D1} & \cdots & s_{DD} \end{pmatrix} = \frac{1}{n-1} X^T X \tag{13.9}$$

The variance in the original axes is s_{11} and $_{22}$. The covariance matrix in our toy example is:

$$\mathsf{S} = \begin{pmatrix} 9.7 & 8.0 \\ 8.0 & 7.7 \end{pmatrix}$$

Step 2: Project data onto a new axis We'll define the new axis by the **unit vector $\boldsymbol{p}$** (Figure 13.8). The projection of a data point $\boldsymbol{x}_i$ onto this axis (its **component score**) is (as per Equation 13.1)

$$t_i = \boldsymbol{p}^T \boldsymbol{x}_i = p_1 x_{i1} + p_2 x_{i2} \tag{13.10}$$

Step 3: Compute variance in these axes The definition of the variance of the $t^{(1)}$ is:

$$s_t^2 = \frac{1}{n-1} \sum_{i=1}^{n} t_i^2 \tag{13.11}$$

Listing 13.1: Listing of PCA. We are assuming the existance of helperfunctions `standardize()` and `sort_eigenvalues()`.

```
import numpy as np

def standardize(X)...

def sort_eigenvalues(lambda, P)...

def pca(X):
    """Given a data matrix X with n rows and D columns,
       return principal components (P) and
       eigenvalues (lambda)"""
    # Standardise the data X
    Z = standardize(X)
    # Compute the covariance matrix S
    S = np.cov(Z)
    # Find unsorted eigenvectors (P) and eigenvalues (lambda)
    lambdas, P = np.linalg.eig(S)
    # Sort the eigenvectors and eigenvalues
    # in order of largest eigenvalues
    lambdas, P = sort_eigenvalues(lambdas, P)
    # The eigenvalues (lambdas) are proprtional
    # to the amount of variance explained
    for i in range(len(lambdas)):
        print('PC' + str(i+1) + ' explains ' +
        str(round((lambdas[i] / np.sum(lambdas))*100)) +
        '% of the variance.')
    return(lambdas, P)
```

If we substitute Equation 13.10 into this equation we get the following:

$$\begin{aligned} s_t^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(p_1 x_{i1} + p_2 x_{i2})^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(p_1 p_2)\begin{pmatrix} \sum_i x_{i1}x_{x1} & \sum_i x_{i1}x_{x2} \\ \sum_i x_{i2}x_{x1} & \sum_i x_{i2}x_{x2} \end{pmatrix}\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \\ &= \boldsymbol{p}^T \boldsymbol{S}\boldsymbol{p} \end{aligned} \tag{13.12}$$

Our old friend the covariance matrix has reappeared. Although we've demonstrated this in 2 dimensions, the equation is still valid in D dimension.

Step 4: Find direction of axis that maximises variance of data projected onto it (1st principal component, PC 1) We now have an expression for the variance in the component scores for any direction of $\boldsymbol{p}$ we now want to find the direction of $\boldsymbol{p}$ that maximises that variance. We have a constraint that $\boldsymbol{p}$ is of unit length, so $|\boldsymbol{p}| = 1$.

This is a constrained optimisation problem, which we can solve using Lagrange multipliers and differentiation. We won't show the details here, but the result is the following equation:

$$\mathsf{S}\boldsymbol{p} = \lambda\boldsymbol{p}$$

Hopefully you recognise this equation. Its solutions are:

1. $\lambda = \lambda_1$, $\boldsymbol{p} = \boldsymbol{e}_1$, where λ_1 is the biggest **eigenvalue** of S and $\boldsymbol{e}_1$ is the associated **eigenvector**

2. $\lambda = \lambda_2$, $\boldsymbol{p} = \boldsymbol{e}_2$, where λ_2 is the second biggest **eigenvalue** of S and $\boldsymbol{e}_2$ is the associated **eigenvector**

We choose the **first principal component** $\boldsymbol{p}_1$ to be the eigenvector $\boldsymbol{e}_1$ with the largest eigenvalue λ_1. λ_1 is the variance of the 1st component scores $\boldsymbol{t}^{(1)}$.

Step 5: Interpret Finding the eigenvalues and eigenvectors for our example, we arrive at the first principal component being:

$$\boldsymbol{p}_1 = \begin{pmatrix} 0.75 \\ 0.66 \end{pmatrix} \begin{array}{l} \text{Java} \\ \text{Coffee} \end{array} \qquad \lambda_1 = s^2_{t^{(1)}} = \boldsymbol{p}_1^T \mathsf{S}\boldsymbol{p}_1 = 16.5$$

The first component score $\boldsymbol{t}^{(1)}$ is the "Java-coffeeness" of a student.

Step 6: Find the 2nd principal component (PC 2) In 2D our job is already done, since there is only one direction perpendicular to $\boldsymbol{p}_1$, and eigenvectors (and therefore principal components) are always orthogonal to each other. It's the other eigenvector of S, $\boldsymbol{p}_2 = \boldsymbol{e}_2$, with eigenvalue λ_2, which is the variance of the 2nd component scores $\boldsymbol{t}^{(2)}$.

In D dimensions, the principal components are the D eigenvectors of the $D \times D$ matrix S. It's helpful to introduce more matrix notation here. We arrange the principal components in the **rotation matrix**:

$$\mathsf{P} = \begin{pmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 \end{pmatrix}$$

Step 7: Quantify what is lost by dimensionality reduction We can reverse the transformation from the scores to the original data

$$\mathsf{X} = \mathsf{T}\mathsf{P}^T$$

If we drop the 2nd PC from P and the 2nd PC scores from T, we can reconstruct a 1-dimensional version of the original data:

$$\tilde{\mathsf{X}}^{(1)} = \boldsymbol{t}^{(1)}\boldsymbol{p}_1^T$$

We can see (Figure 13.9) that the first principal component (the "Java-coffeeness") score of a student tells us **a lot** about them – but how much? Consider the **total variance**, the sum of the variances of the data:

$$\sum_{i=1}^{D} s_i^2 = \sum_{i=1}^{D} \lambda_i$$

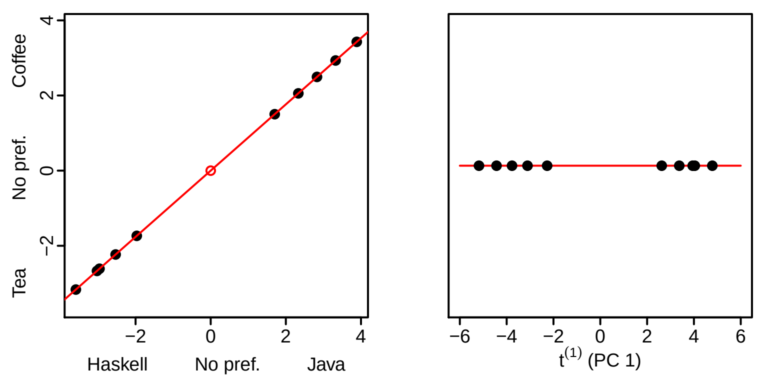


Figure 13.9: Projection into the original space.

It is equal to the sum of the eigenvalues of the covariance matrix. Thus the fraction of the total variance "explained" by the ith principal component is:

$$\frac{\lambda_i}{\sum_{j=1}^{D} \lambda_j}$$

In our toy example,

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{16.5}{16.5 + 0.61} = 96\%$$

Thus we can now be more precise about how much the first principal component (the "Java-coffeeness") score of a student tells us about them: 96% of of the variance.

Related Python Lab: PCA

`https://github.com/Inf2-FDS/FDS-S1-08-pca`

- Write PCA from scratch
- Use PCA on dataset using sklearn
- Visualise data with PCA

Part IV

Statistical inference

Chapter 14

Randomness, sampling and simulation

Recommended reading

- *Modern Mathematical Statistics with Applications*, Sections 6.1 and 6.2

14.1 Introduction to statistical inference

Inferential statistics In the chapter on Descriptive statistics, we looked at the difference between a sample and a population. We also considered a number of statistics that could apply to the population and to the sample: the mean, variance, standard deviation and median. **Statistical inference** is the process of drawing conclusions about quantities that are not observed (Gelman et al., 2004).

One example of an statistical inference task is **estimation** of a population statistic from a sample of that population. For example, suppose we've weighed a sample of 10 wild cats from a population of 400. We know what the sample mean and sample standard deviation is. On the basis of this information, what is the best estimate of the population mean (**point estimation**), and how confident can we be in that estimate (**confidence interval estimation**)?

Inferential statistics has been around for much longer than the word "statistics". The 9th-century book "Manuscript on Deciphering Cryptographic Messages" written in Arabic by Al-Kindi (educated in Baghdad) shows how to decipher encrypted messages by frequency analysis, i.e. counting the frequency of particular letters.

Inferential statistics tasks Inferential statistics can seem like a toolbox full of tools with confusing names such as "standard error of the mean", t-test, χ^2 test, bootstrap, and a confusing set of rules about what to use each tool for. We're going to try to give you an idea of what task each tool is useful for, and how it works. There are three main tasks we will consider:

1. Estimation
2. Hypothesis testing
3. Comparing two samples (A/B testing)

Estimation We've already given one example of estimating an unobserved quantity (the population mean). Another example of an unobserved quantity is linear regression coefficients. We already know how to find (point) estimates of them, using the formulae we covered earlier in the course. But in part of the course we will learn how to estimate the **confidence intervals** around the point estimates, which will tell us how much uncertainty there is in our estimates. For example, we'll be able to say that we estimate the mean weight of squirrels in the

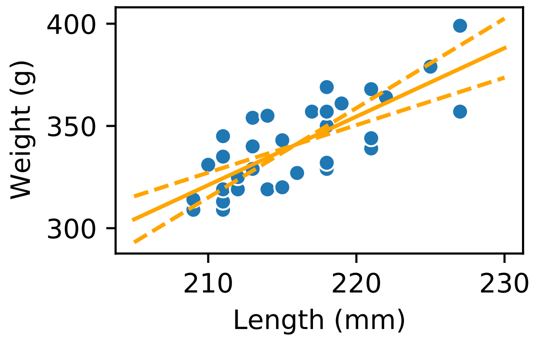


Figure 14.1: Uncertainty in regression line. The best estimate regression line (solid) and the lines at the edge of the 95% confidence interval (dashed lines). Note this plot is simplified, since the uncertainty in the intercept is not represented.

population to be 320 ± 16g, with 95% confidence, i.e. in the confidence interval $[304, 336]$g. In a linear regression of the weight of a sample of squirrels on their length (Figure 14.1), we will be able to say that the best estimate of the slope of the regression line is 3.35 g/mm, but we are 95% confident that the slope lies in the interval [2.32, 4.38] g/mm.

Hypothesis testing In hypothesis testing, we are trying to ascertain which of two or more competing theories are the best explanation of the data. For example, in 1965 a court case was brought against the state of Alabama (Swain versus Alabama, 1965) due to there being no Black members of the jury in a trial.[1] Part of the case concerned the fact that at one stage of the jury selection, 8 Black people were chosen for a jury panel of 100 people, but the fraction of Black people in the population was 26%. Our question is "Is the jury selection system biased against Black people?".

Comparing two samples (also known as A/B testing) Here we have two samples that have been treated differently, and we want to either test if the groups are different, or estimate how different they are. For example, to find out the effectiveness of a vaccine, we select a sample of volunteers from the population randomly, divide them randomly into two groups, give the vaccine to one group (Treatment group) and give the other group a placebo (Control Group). In the vaccine group 3 volunteers catch the disease, but in the placebo group 95 volunteers catch the disease. Is the vaccine effective? How much would we expect the vaccine to cut the risk of catching the disease if we give it to the whole population?

In the context of user testing, often in web applications, this is called A/B testing. A famous example was at Amazon, where a developer had the idea of presenting recommendations as customers add items to a shopping cart (Kohavi et al., 2007). Amazon managers forbid the developer to work on the idea, but the developer disobeyed orders and ran a controlled experiment on customers, by splitting them into two groups ("A" and "B"), one which had recommendations shown and one which didn't. The group which had recommendations shown bought more, and displaying recommendations quickly became a priority for Amazon.

Two approaches to statistical inference We are going to learn a number of techniques for undertaking point and interval estimation, hypothesis testing and comparing samples. We will also think carefully about the interpretation of these techniques. There are two main approaches to undertaking statistical inference tasks, statistical simulations and statistical theory:

1. **Statistical simulations**: Here we use repeated random sampling to carry out the statistical inference procedures. The advantages of statistical simulation procedures are they often require fewer assumptions about the data and/or hypothesis than statistical theory, and they require somewhat less theory to understand. However, they can be compute-intensive, and care is still needed in their use.

[1]We found the example of Swain versus Alabama in Adhikari et al. (2020), and follow their treatment.

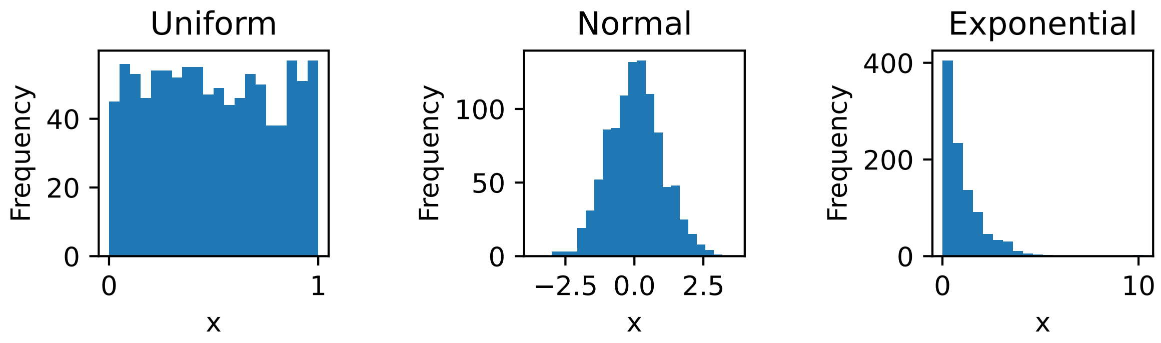


Figure 14.2: Histograms of 1,000 samples taken from normal, uniform and exponential distributions.

2. **Statistical theory**: Here we use the properties of various well-known theoretical distributions to draw inferences about our data. We need to check that the assumptions behind the distribution match the statistical question we are trying to answer. For example, a distribution of delays to flights is likely to be highly right-skewed, so we shouldn't assume a normal distribution when dealing with it. Typically, the process is not compute-intensive: very often it amounts to arithmetic and then reading of a quantity from a distribution table. These procedures come as standard in a number of stats packages, including R and Python's statsmodels.

A number of fundamental concepts underpin both the statistical theory and statistical simulations.

Plan for this part of the course The plan for the statistical inference topics in this part of the course will be:

1. Fundamental theory (the rest of this chapter):
   - We'll learn how we can use statistical simulations to generate samples from a model and compute statistics for each of these samples to give a **sampling distribution**.
   - Learn about the distribution of the mean of repeated samples from a model. This will lead us to the central limit theorem, which can help us to estimate the uncertainty in our estimate of the mean, i.e. confidence intervals, and the law of large numbers, which also helps with estimation.
2. Estimation and Confidence intervals
3. Hypothesis testing and p-values
4. A/B testing

14.2 Sampling, statistics and simulations

Sampling A prerequisite for statistical simulations is being able to sample from probability distributions and from sets of discrete items, including observed data.

Random sample In a **random sample** of size n from either a probability distribution or a finite population of N items, the random variables $X_1, \ldots, X_n$ comprising the sample are all independent and all have the same probability distribution.

Sampling from probability distributions You should be familiar with sampling from random number generation. A standard random number generator produces numbers within an interval (e.g. [0, 1]) with uniform probability for each number, i.e. it samples from a uniform distribution. We can demonstrate the distribution of a standard random number generator by drawing many samples and plotting a histogram (Figure 14.2). We adapt these functions to sample from any univariate distribution, e.g. a normal distribution or an exponential distribution (Figure 14.2).

Sampling from a set of discrete items We can also sample from a population of discrete items. We can select n items from a set of N items at random either **without replacement** or **with replacement**. If we sample without replacement, it is as though we are pulling items of various types (e.g. coloured balls) at random out of a bag, and not replacing them. We can only sample up to N items, and also, as we remove items from the bag, the probabilities of drawing a particular type (colour) changes. If we sample with replacement, we put the item back in the bag, before making our next choice – we can carry on doing this for ever. We could construct an algorithm for random sampling either with or without replacement from a uniform random number generator, but these functions are provided in packages such as `numpy.random.choice` in Python.

A particular application of sampling from a set of discrete items is creating a sample of a larger data set.

Non-random samples from a population We can also imagine ways of sampling that are not systematically random. For example, we might have a list of the daily takings in a restaurant. We could take the first n days. But suppose that the dataset has been sorted in terms of takings? We would then have days with low takings at the start of the list, so the statistics of the sample would not resemble the statistics of the population. We could try taking every 7th day in the list – but if the list is in date order we will always be sampling from one day of the week, e.g. Mondays. Random sampling ensures that we don't have this type of problem.

Samples of convenience When we are collecting data, it might be tempting to sample from the data that we can collect conveniently. For example, a polling company may find it easier to contact people who have more time to answer the phone, which may tend to be retired people. If we don't correct for this sort of bias, it's called a **sample of convenience**. One way of combating convenience sampling is **stratified sampling**, in which the sampling is targeted so that the proportions of attributes of the sample matches the proportions in the population.

Definition of a statistic Before going further, it's helpful to have the definition of **statistic**: "A **statistic** is any quantity whose value can be calculated from sample data."(*Modern Mathematical Statistics with Applications* 6). We probably recognise the mean, variance and median as statistics by this definition. But we've also derived other quantities from sample data, such as the correlation coefficient and regression coefficients – they are also statistics. We will follow *Modern Mathematical Statistics with Applications* and denote a statistic using an uppercase letter, to indicate that it is a random variable, since its value depends on the particular sample selected. E.g. $\overline{X}$ represents the mean and S^2 the variance.

Simulations and sampling Before considering inferential statistics proper, we will focus on running statistical simulations, i.e. using a computer program to make predictions from probabilistic models of real-world processes. For example, the probabilistic model of tossing a coin multiple times is that the tosses are independent and that the probability of a head is 1/2 (or perhaps another value, if with think the coin is loaded). The statistical simulation generates a sequence of heads and tails.

To do this we need to decide on:

- The statistic of interest ($\overline{X}$, S, etc.)
- The population distribution (e.g. normal with particular mean and variance) or set of discrete items
- The sample size (denoted n)
- The number of replications k

The simulation procedure is then:

1. For i in $1, \dots, k$
   (a) Sample n items from the population distribution or set of discrete items
   (b) Compute and store the statistic of interest for this sample
2. Generate a histogram of the k stored sample statistics

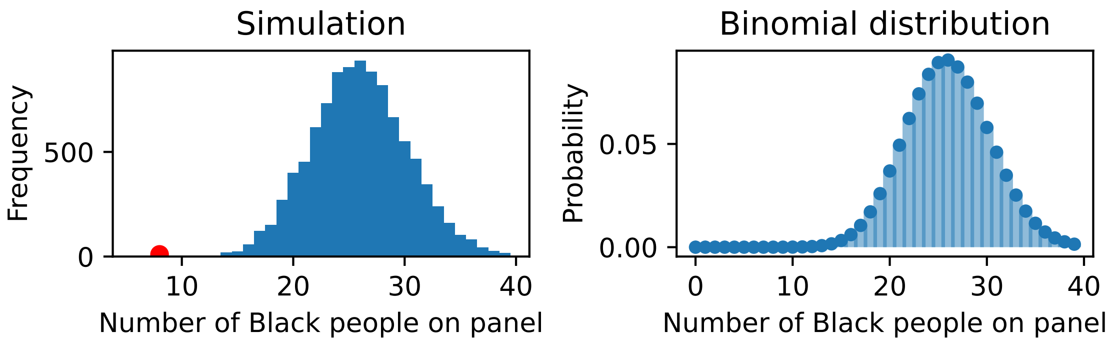


Figure 14.3: Results of statistical simulations of the panel size in Swain versus Alabama (1965). The blue histogram shows how many of 10 000 simulations produced jury panels of 100 with the given number of Black people on them. The red dot indicates the number of Black jurors in Swain versus Alabama (1965).

Example of hypothesis testing using a simulation experiment To demonstrate the utility of the statistical experiment we've introduced, let's look again at the example in which 26% of the population is Black and 8 Black people are selected to be on a jury panel of 100 people. The null hypothesis H_0 is "The jury panel was chosen at random from the population". We can map the null hypothesis onto the general framework above as follows:

- The statistic of interest is T_0, the number of Black people in a sample of $n = 100$ panel members
- The population distribution is a Bernoulli distribution with the sample space Black, Non-Black in which $p(\text{Black}) = 0.26$.
- The sample size is $n = 100$
- The number of replications $k = 10\,000$

We follow the procedure described in the previous section to give the results shown in Figure 14.3. Coding this up will be an exercise for you in the Labs. We can see that none of the 10 000 simulations of the null hypothesis produced a jury with 8 members, suggesting that we should reject the null hypothesis in favour of an alternative one. This looks like a clear-cut case; in the topic on Hypothesis testing, we'll consider in more detail how to interpret the results when the data is less distinct from the simulations.

Deriving the sampling distribution Note that in this example, we didn't have to go to the trouble of running a simulation experiment. We might have noticed that the total number of Black people will be distributed according to a binomial distribution with $n = 100$ and $p = 0.26$.

14.3 Distributions of statistics of small samples from probability distributions

Example of sampling from probability distributions In the previous example, we've sampled a total number of successes from a Bernoulli distribution. We'll now look at what happens when we sample the mean, standard deviation and median from the normal, uniform and exponential distributions by running the following simulations:

- Statistics of interest: mean $\overline{X}$, standard deviation S and median $\tilde{X}$
- Population distribution: Normal distribution with mean 0 and variance 1, Uniform distribution on [0, 1], Exponential distribution $p(x) = e^{-x}$.
- Sample size $n = 10$
- Number of replications $k = 10,000$

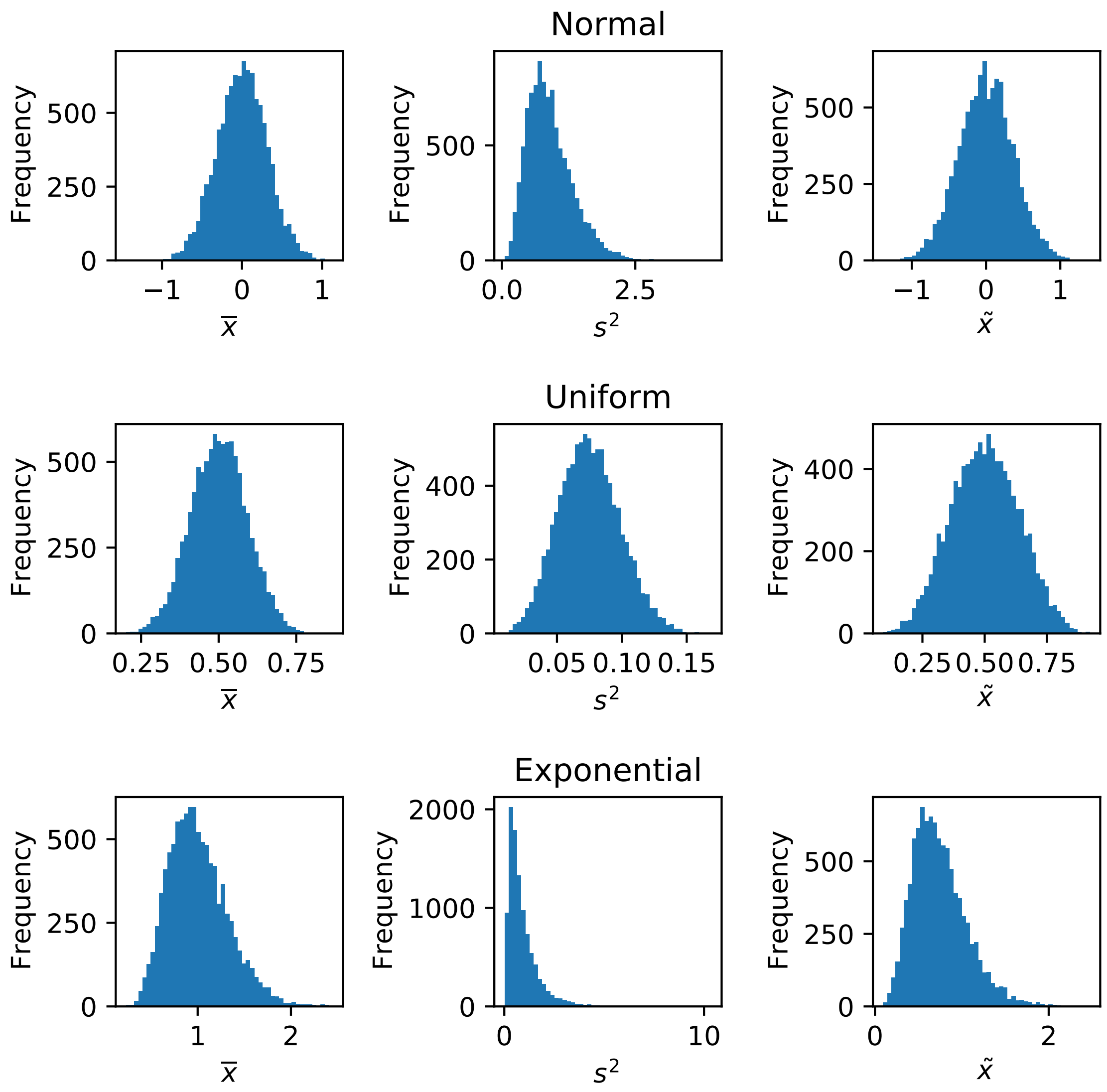


Figure 14.4: Sampling distribution generated by 10,000 simulations of the mean $\overline{x}$, variance s^2 and median $\tilde{x}$ of 10 samples drawn from a normal distribution (top row), uniform distribution (middle row) and exponential distribution (bottom row).

Figure 14.4. There are a number of points to notice about this plot:

Sample mean (first column), all distributions The distribution of the mean is narrower than the original distribution in every case. This is because some of the variability in the individual samples is averaged out. The standard deviation of this distribution is called the **standard error of the mean**.

Sample mean of normal distribution The distribution looks to be normal – it turns out that this is easy to prove.

Sample mean of uniform distribution The distribution is symmetric and looks to be near-normal.

Sample mean of exponential distribution The distribution is clearly skewed, but less so than the original exponential distribution.

Sample variance (second column) All these distributions are skewed, reflecting the fact that it's very unlikely to get 10 samples that are all very close together, and therefore have low variance. It turns out that there is a theoretical distribution (the χ^2 distribution) that describes the shape of sample variance from the normal distribution.

Median (third column) The main point to draw from this column is that we can use the simulation method to produce a distribution for any statistic, regardless of how easy it would be to calculate a theoretical distribution for it.

As we will see later, we could generate the sampling distribution of the mean and the variance analytically rather than by simulation. However, it is not always possible to compute sampling distributions of the desired statistics analytically, and we can always run statistical simulations.

14.4 The distribution of the sample mean of large samples

The distribution of the sample mean A particularly common statistic of interest is the sample mean. It therefore makes sense to understand how the distribution of the sample mean depends on the distribution from which we sample and the number of samples we take.

We've already seen in Figure 14.4 that the sampling distribution of the mean of 10 items from a normal distribution is itself a normal distribution, though with smaller variance. However, the sample mean distributions for an exponential distribution in our simulation, was not normal. We can repeat the simulation experiments for the three distributions, but with larger sample sizes of $n = 1000$ and $n = 10000$ (Figure 14.5). What we see is remarkable: *the distributions of the sample means are all normal, regardless of whether they came from a normal, uniform or exponential distribution*. Perhaps less remarkably, we also see that as the sample size gets larger the distributions get narrower.

These simulations and observations give us the intuition for two very important statistical laws that apply to many non-normal distributions, as well as normal ones:

- The Central Limit Theorem
- The Law of Large Numbers

Central Limit Theorem Here is an informal statement of the **Central Limit Theorem** (CLT):

> The distribution of the mean or sum of a random sample of size n drawn from any distribution will converge on a normal distribution as n tends to infinity.
>
> In the case of the sample mean, its expected value is the same as the mean of the population distribution, and its expected variance is a factor of n lower than the population variance.
>
> In the case of the sample sum, its expected value is the same as the product of the sample size n and the expected value of the distribution, and its expected variance is n times the variance of the population distribution.

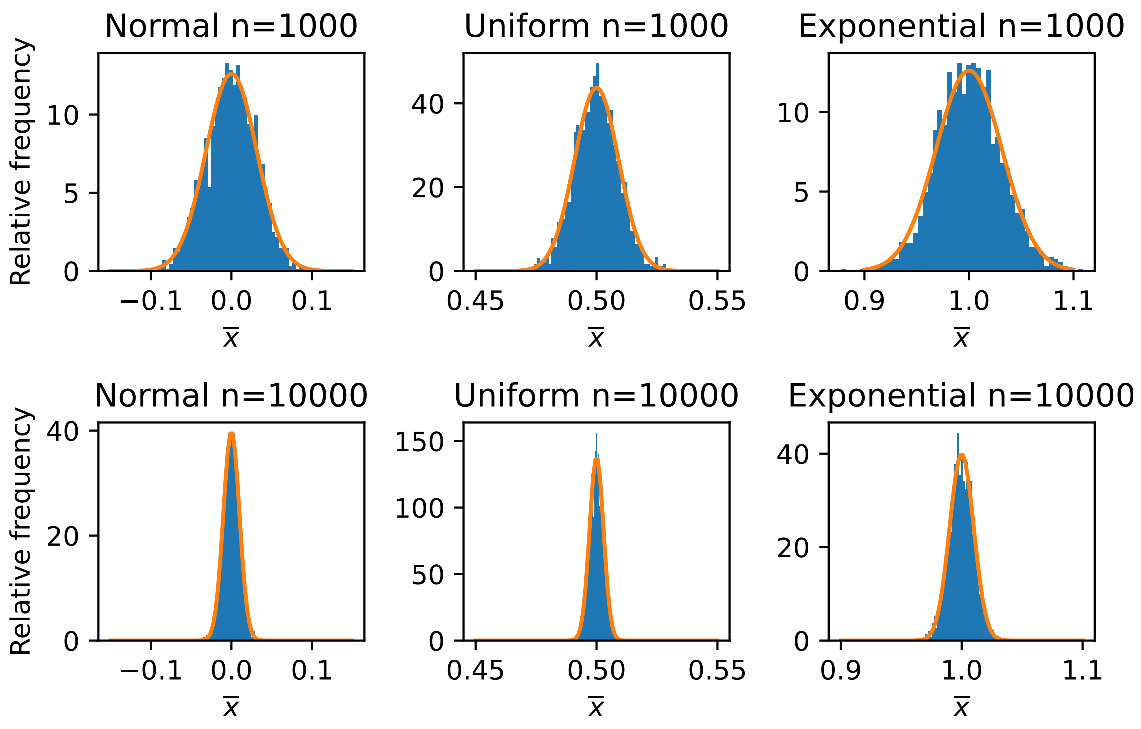


Figure 14.5: Distributions of means from samples of size $n = 1000$ (top row) and $n = 10000$ (bottom row) drawn from the normal, uniform and exponential distributions shown in Figure 14.2. The blue histograms show the histograms obtained from $k = 2000$ simulations. The orange curves are normal distributions with mean equal to the mean of the original distribution and variance $\sigma_{\overline{X}}$ equal to σ^2/n, where σ^2 is the variance of the original distribution.

We denote the expected variance of the mean $\sigma^2_{\overline{X}}$ and we call the standard deviation of the mean $\sigma_{\overline{X}}$, or the **standard error of the mean**, often abbreviated as SEM. It's important to note that the SEM is *not* the same as the standard deviation of the original distribution. According to the statement above, an estimate of the SEM is $\hat{\sigma}_{\overline{X}} = \sigma/\sqrt{n}$.

We can verify that this statement holds in the case of sampling a mean in Figure 14.5 by computing the means and SEM from the simulations and comparing with the expected values of μ (population mean) and $\sigma_{\overline{X}} = \sigma/\sqrt{n}$.

The Swain versus Alabama jury selection example demonstrates the CLT applied to a total $T_0 = \sum_{i=1}^{n} X_i$, where $X_i = 1$ indicates a Black member of the population was selected, and $X_i = 0$ indicates non-Black. The distribution is a Bernoulli distribution with population mean $\mu = p = 0.26$, the probability of picking a Black person. We can see from Figure 14.3 that the mean of the total is $n\mu = 100 \times 0.26 = 26$, and the variance is approximately $\sigma^2_{T_0} \approx n\sigma^2 = np(1-p) = 19.24$, as expected for a Bernoulli distribution, giving a standard deviation of 4.38. Furthermore, the distribution is approximately normal.

The law of large numbers Here is an informal statement of the **law of large numbers**:

> In the limit of infinite n, the expected value of the sample mean $\overline{X}$ tends to the population mean μ and the variance of the sample mean $\overline{X}$ tends to 0.

Note that sometimes the law of large numbers is referred to as the "law of averages". This can lead to confusion. The law of averages is sometimes called the "Gambler's fallacy", i.e. the idea that after a run of bad luck, the chance of good luck increases. If the events that are being gambled on are independent of each other (e.g. successive tosses of the same coin), the probability of a head will be the same regardless of how many tails have preceded it.

In the second row of Figure 14.5 we can see that the distribution for $n = 10000$ is narrower than the distribution for $n = 1000$, and that the sample means converge on the population means. The law of large numbers says that we could, in principle, continue this process by choosing an n as large as we would like to make the variance as small as desired.

Formal statement of the central limit theorem (*Modern Mathematical Statistics with Applications* 6.2) Let $X_1, \dots, X_n$ be a random sample from a distribution with mean μ and variance σ^2. Then, in the limit $n \to \infty$ the standardised mean $((\overline{X} - \mu)/(\sigma/\sqrt{n}))$ and standardised total $((T_0 - n\mu)/(\sqrt{n}\sigma))$ have a normal distribution. That is

$$\lim_{n\to\infty} P\left(\frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \leq z\right) = P(Z \leq z) = \Phi(z)$$

and

$$\lim_{n\to\infty} P\left(\frac{\overline{T_0} - n\mu}{\sqrt{n}\sigma} \leq z\right) = P(Z \leq z) = \Phi(z)$$

where $\Phi(z)$ is the cumulative distribution function (cdf) of a normal distribution with mean 0 and s.d. 1. Thus, when n is sufficiently large, $\overline{X}$ has an approximately normal distribution with mean $\mu_{\overline{X}} = \mu$ and variance $\sigma^2_{\overline{X}} = \sigma^2/n$ and the distribution of T_0 is approximately normal with mean $\mu_{T_0} = n\mu$ and variance $\sigma^2_{T_0} = n\sigma^2$. We can also say that the standardised versions of $\overline{X}$ and $\overline{T_0}$ are **asymptotically normal**.

Formal statement of the (weak) law of large numbers (*Modern Mathematical Statistics with Applications* 6.2)

Let $X_1, \dots, X_n$ be a random sample from a distribution with mean μ and variance σ^2. As the number of observations n increases, the expected value of the sample mean remains $E[\overline{X}] = \mu$, but the expected variance $V[\overline{X}] = E[(\overline{X} - \mu)^2] \to 0$. We say that "$\overline{X}$ converges in mean square to μ".

More formally, the probability that the difference between the sample mean and population mean is greater than an arbitrary value ϵ is

$$P(|\overline{X} - \mu| \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2}$$

for any value ϵ. Thus, as $n \to \infty$, the probability approaches 0, regardless of the value of ϵ.

A proof of this statement relies on **Chebyshev's inequality**, and can be found in *Modern Mathematical Statistics with Applications* 6.2.

Note that this is the statement of the weak law of large numbers. There is also a strong law, which has somewhat more stringent requirements on convergence. All distributions that obey the strong law also obey the weak law, but some distributions only obey the weak law and some obey neither law. A discussion of this topic is beyond the scope of this course; the distributions that do not obey the distribution tend to be “weird”, e.g. having infinite variance.

Frequentist versus Bayesian statistics

You may have heard of the difference between Frequentist and Bayesian statistics. The two systems have different philosophical bases, but, in simpler cases, often end with similar results. Roughly speaking, the differences between the two are:

Frequentist The population is a fundamental concept. There is just one possible value of the population mean and variance, i.e. the one that exists in the population. In estimation, we are trying to estimate these quantities, and in hypothesis testing, we are trying to compare our sample with this population.

Bayesian A fundamental concept is the model of the likelihood of the data given parameters (such as the mean). The parameters themselves are uncertain. Conceptually, the population itself is generated from the model, so a number of combinations of parameters and luck may have generated the particular value of (say) the mean observed in a population. Before we have seen any data, we have an initial idea about the distribution of the parameters (the prior). The inference process involves using the data to update this prior distribution to give a distribution of the parameters given the data.

For around a century, there has been controversy about which approach is best. Broadly speaking, we will be using Frequentist approaches in this course. At the level we are working at here, it will give very similar results to Bayesian approaches. The important thing is to understand the meaning and interpretation of our inference.

Related Python Lab: Randomness, sampling and simulations

`https://github.com/Inf2-FDS/FDS-S1-10-randomness-sampling-simulations`

Chapter 15

Estimation

Recommended reading

- *Modern Mathematical Statistics with Applications*, Sections 7.1

15.1 Point estimation

Estimation In the chapter on Descriptive statistics, we made the distinction between populations and samples. In many cases we may want to know a property of the population, and it may be only feasible to collect data for a sample of that population. For example, we may wish to know the mean (or median) weight of the population of Scottish wildcats, or the voting intentions of a population. In these examples a population of finite size exists, even if we don't know exactly how large the population is.

There are other populations that are less well-defined: for example the weight of cheese added to a pizza by a pizza-chef varies from pizza to pizza. The population of pizzas created by the chef is never really complete – what we are interested in is the mean and standard deviation of the distribution of the weight of cheese that chef adds to the pizzas. It seems reasonable to assume that the cheese weight is normally distributed, since (a) it is a continuous quantity and (b) there is no particular reason to think that the distribution is skewed. The normal distribution is a model of the data.

Parameters In both of these examples we call the population mean and standard deviation **parameters**. In the case of the pizza chef, the mean and standard deviation are parameters of normal distribution, which describe its centre and width. In some distributions, e.g. an exponential distribution $p(x) = \exp(-\lambda x)$ (for $x > 0$), the mean and standard deviation are not separate parameters; they are both equal to the inverse of the parameter λ.

Estimation problems The Oxford English dictionary defines **estimation** as "the process of forming an approximate notion of (numbers, quantities, magnitudes, etc.) without actual enumeration or measurement." In other words, we would like to get an approximate idea of population parameters without looking at the entire population. There are two main estimation problems:

1. What is the best way of using the sample to construct a **point estimator** for each population parameter?
2. How do we construct a **confidence interval** to indicate how accurate we expect that point estimator to be?

The answers to these questions will depend on the distribution of the data, and is quite a complex area. We will give an overview of some of the issues here, but not go into depth.

Generic notation for parameters and estimators To help with making some definitions general, we will refer to a generic parameter by the Greek letter ϑ and its estimator by $\hat{\vartheta}$. We'll also use the hat notation for the specific parameters. For example, $\hat{\mu} = \overline{X}$ means "the point estimator of the population mean μ is the sample mean $\overline{X}$".

More than one estimator for a parameter In some cases, we can have more than one estimator for a parameter. For example, both the mean and the median are estimators of the centre μ of a symmetric distribution, so we can write $\hat{\mu} = \tilde{X}$ as well as $\hat{\mu} = \overline{X}$.

Capture-recapture method and estimator

Suppose we want to estimate the number of squirrels N in a population. We can do this with a clever method called capture-recapture:

1. Capture n of the squirrels, tag them so that they can be identified if caught again, then release them.
2. Wait for the squirrels to move around.
3. Recapture K of the squirrels and record the number k of these recaptured squirrels that have tags.
4. The estimator of the number of squirrels in the population is

$$\hat{N} = \frac{nK}{k}$$

This should work if the capturing and recapturing processes are random. If this is the case, the expected proportion of tagged squirrels in the whole population n/N is equal to the proportion in the recaptured sample k/K, hence the estimator.

15.2 Estimation bias and variance

Bias and variance Estimators have two properties: bias and variance, which we've summarised graphically in Figure 15.1.

In general, when we estimate a parameter ϑ using an estimator $\hat{\vartheta}$ we will be wrong. We define the error of any particular estimation as $\hat{\vartheta} - \vartheta$. We define the **bias of an estimator** as[1]:

$$\text{bias} = \mathrm{E}[\hat{\vartheta} - \vartheta] = \mathrm{E}[\hat{\vartheta}] - \vartheta \tag{15.1}$$

The bias tells us, on average, by how much the estimate is too high or too low. If, on average, an estimator gets the right result, i.e. $\mathrm{E}[\hat{\vartheta}] - \vartheta = 0$ for *any* value of ϑ we say that the estimator is **unbiased**.

Mean squared error of an estimator We would like each estimate of the parameter to be as close to the true value of the estimator as possible. A measure of close we can expect estimates to be to the true value is the **mean squared error of an estimator**, which is defined as:

$$\text{MSE} = \mathrm{E}[(\hat{\vartheta} - \vartheta)^2] \tag{15.2}$$

It turns out that the MSE can be decomposed into the variance of the estimator and the squared bias[2]:

$$\text{MSE} = \mathrm{E}[(\hat{\vartheta} - \vartheta)^2] = \mathrm{V}[\hat{\vartheta}] + (\mathrm{E}[\hat{\vartheta}] - \vartheta)^2 = \text{variance} + (\text{bias})^2 \tag{15.3}$$

We might think that we should always prefer unbiased estimators that have minimal MSE (or, equivalently zero bias and minimal variance). It turns out that for some parameters of some distributions the unbiased estimators do not have minimal MSE, and that by adding bias we can reduce the variance, thereby reducing the MSE. For an example, see Example 7.4 in *Modern Mathematical Statistics with Applications*.

[1]The notation $\mathrm{E}[X]$ means "expected value of the random variable X". See Section 4.2 of *Modern Mathematical Statistics with Applications* for a definition.

[2]The notation $\mathrm{V}[X]$ means "(expected) variance of the random variable X". See Section 4.2 of *Modern Mathematical Statistics with Applications* for a definition.

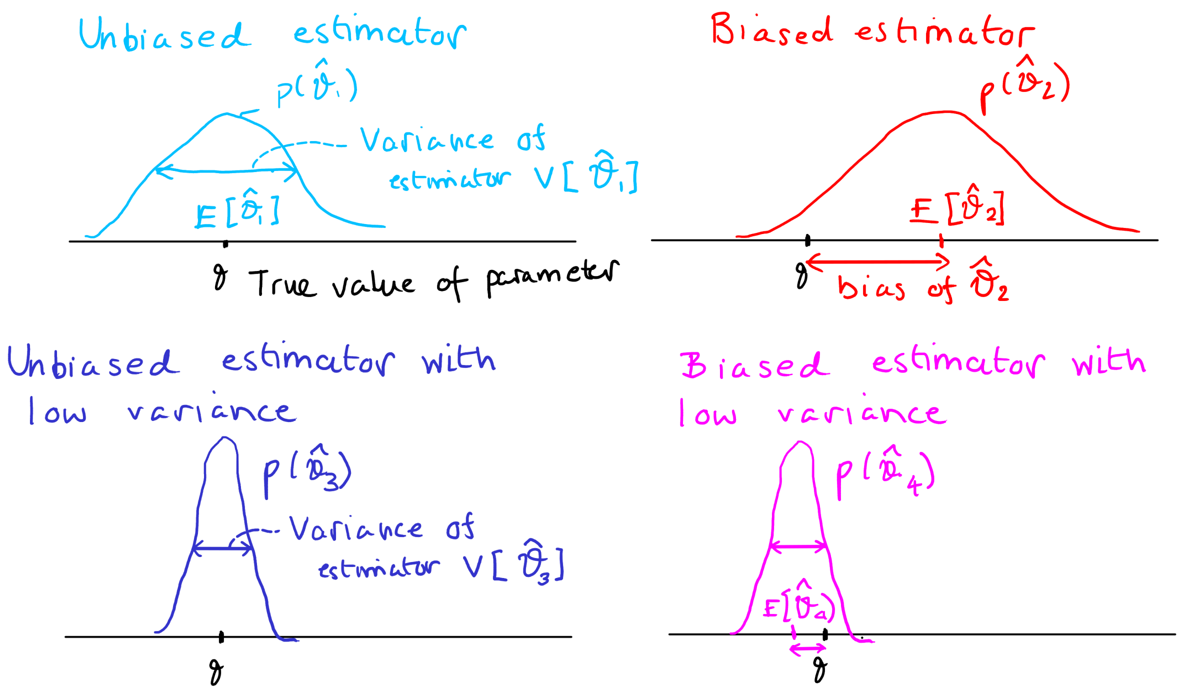


Figure 15.1: Schematic diagram of bias and variance. In each plot the true value of the parameter is indicated by the black point ϑ. The estimator $\hat{\vartheta}_1$ (top left, light blue) is an unbiased estimator, meaning that its expected value is equal to ϑ; the distribution is centred on the true value ϑ. In the top right plot (red) ϑ_2 is a biased estimator; its expected value $E[\hat{\vartheta}_2]$ is greater than the true value, and the difference between the two is the bias. In the bottom left plot, the distribution of the estimator $\hat{\vartheta}_3$ (dark blue) is unbiased and has lower variance, meaning that it is more tightly clustered on the true value. In the bottom right plot, $\hat{\vartheta}_4$ is a biased estimator, but has lower variance than $\hat{\vartheta}_1$. It may be preferable, as it has lower mean squared error (MSE).

Example: point estimators for the mean of a normal distribution with known variance We'll first consider the simplest, and rather artificial case, namely a random sample of n observations $X_1, \dots, X_n$ from a normal distribution in which we *know* the variance parameter σ^2 independently of the data. An obvious choice for a point estimator of the mean parameter μ is the sample mean, i.e. $\hat{\mu} = \overline{X}$. For a normal distribution, for *any* value of n, the standardised variable $(\overline{X} - \mu)/(\sigma/\sqrt{n})$ has a standard normal distribution (this can be proved quite easily). From this we can derive that $\mathrm{E}[\overline{X}] - \mu = 0$, which means that the bias is zero. We can also see that

$$\mathrm{MSE} = \mathrm{E}[(\overline{X} - \mu)^2] = \sigma^2/n \tag{15.4}$$

Since the bias is 0, the variance is equal to the MSE.

We can see that as we increase the number of samples n, the MSE decreases, which makes sense.

Example of a senseless biased estimator Note that an estimator does not have to be unbiased or have minimal variance. For example, we could try to estimate the mean with $\overline{X} + 1$. There would then be a bias of 1 and the MSE would be higher. This particular addition of bias makes no sense, but there are cases (see *Modern Mathematical Statistics with Applications*, Section 7.1) where it can make sense.

Example of a biased estimator from Machine Learning As noted in the section on Cross-validation, cross-validation can be used to estimate the value of a metric (e.g. accuracy) when a classifier is tested on unseen data. However, if the cross-validation data has been used to choose a hyperparameter, the cross-validated estimate of the metric is biased. We can regard the value of the metric measured from unseen data as the quantity being estimated ϑ, and the cross-validated value of the metric as the estimator $\hat{\vartheta}$.

Derivation of unbiased estimator of the population variance (not examinable)

We can now understand the reasoning for why the unbiased estimator of the variance σ^2 has an $n-1$ in the divisor (see Why the divisor $n-1$ in the sample variance?) The estimator of the mean is:

$$\hat{\mu} = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{15.5}$$

The naive estimator of the variance would be:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu})^2 \tag{15.6}$$

We rearrange this expression to collect terms that comprise two independent random variables (e.g. X_i and X_j) or one random variable squared (i.e. X_i^2):

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu})^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(X_i - \frac{1}{n}\sum_{i=1}^{n} X_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\frac{n-1}{n}X_i - \frac{1}{n}\sum_{j\neq i} X_j)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)^2}{n^2}X_i^2 - 2\frac{n-1}{n^2}X_i\sum_{j\neq i}X_j + \frac{1}{n^2}\left(\sum_{j\neq i}X_j\right)^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)^2}{n^2}X_i^2 - 2\frac{n-1}{n^2}X_i\sum_{j\neq i}X_j + \frac{1}{n^2}\left(\sum_{j\neq i}X_j^2 + \sum_{j\neq i}\sum_{k\neq j,i}X_jX_k\right)\right)
\end{aligned} \tag{15.7}$$

We then compute the expectation of this estimator and use the properties of expectations to bring it into a form where we can compare it to the actual population variance $\sigma^2 = \mathrm{E}[X^2] - (\mathrm{E}[X])^2$:

$$\begin{aligned}
\mathrm{E}[\hat{\sigma}^2] &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)^2}{n^2}\mathrm{E}[X_i^2] - 2\frac{n-1}{n^2}\mathrm{E}[X_i\sum_{j\neq i}X_j] + \frac{1}{n^2}\sum_{j\neq i}\mathrm{E}[X_j^2] + \frac{1}{n^2}\sum_{j\neq i}\sum_{k\neq i,j}\mathrm{E}[X_jX_k]\right) \\
&= \left(\frac{(n-1)^2}{n^2} + \frac{n-1}{n^2}\right)\mathrm{E}[X_i^2] + \left(-2\frac{(n-1)^2}{n^2} + \frac{(n-1)(n-2)}{n^2}\right)\mathrm{E}[X_i]\mathrm{E}[X_j] \\
&= \frac{n(n-1)}{n^2}\mathrm{E}[X_i^2] - \left(\frac{(n-1)^2}{n^2} + \frac{n-1}{n^2}\right)\mathrm{E}[X_i]\mathrm{E}[X_j] \\
&= \frac{n-1}{n}\left(\mathrm{E}[X_i^2] - (\mathrm{E}[X_i])^2\right) \\
&= \frac{n-1}{n}\sigma^2
\end{aligned} \tag{15.8}$$

Therefore this estimator has a bias:

$$\mathrm{E}[\hat{\sigma}^2] - \sigma^2 = \frac{n-1}{n}\sigma^2 - \sigma^2 = -\frac{1}{n}\sigma^2 \tag{15.9}$$

indicating that it underestimates the true variance, but amount of the underestimate decreases as n gets larger.

15.3 Standard error

Estimated standard error of an estimator In addition to the point estimate, we would like a measure of how far the point estimate may be from the true value of the parameter. Assuming we have an unbiased (or near unbiased) estimator, we already have a measure that tells us this – it's the variance of the estimator $\mathrm{V}[\hat{\vartheta}]$, which will be equal (or nearly equal) to the MSE. To give a measure that is in the same units as the mean, we tend to take the square root of the variance, to give us the **standard error of an estimator**, which we denote $\sigma_{\hat{\vartheta}} = \sqrt{\mathrm{V}[\hat{\vartheta}]}$.

Going back to the example of estimating the mean of a normal distribution with known variance, Equation 15.4 gives the MSE of the estimator. However, the MSE is defined in terms of the population variance σ^2, which above we assumed that we know – but in real life we only have the *estimate* $\hat{\sigma}^2$ from the sample. However, we can replace any parameters in the formula for the variance with their estimates to give the **estimated standard error of an estimator**, which we denote $\hat{\sigma}_{\hat{\vartheta}}$.

Relationship between standard error of the mean and standard deviation of the distribution In the chapter on Randomness, sampling and simulation, we encountered the standard error of a particular estimator, namely the standard error of the mean (SEM), denoted $\hat{\sigma}_{\overline{X}}$. More generally, we could denote the SEM as $\hat{\sigma}_{\hat{\mu}}$, since that implies that we could be using and estimator other than the sample mean to estimate the mean parameter.

It is important to be clear about the difference between the terms "standard deviation" and "standard error of the mean". The standard deviation tells the variability of the population, distribution or sample. If the standard deviation is describing the population or distribution it's a parameter, and we denote it σ; if the standard deviation is derived from the sample, it's a statistic, and we denote it s.

In the artificial case where we know the standard deviation of the population σ independently of the data, we know from the Central Limit Theorem that the standard error is related to the standard deviation and the sample size by:

$$\sigma_{\hat{\mu}} = \frac{\sigma}{\sqrt{n}} \tag{15.10}$$

This relationship shows that to make the estimator twice as accurate (i.e. halving the SEM) we need to quadruple the size of the data n. Figure 15.2 uses statistical simulations to demonstrate that the SEM behaves as predicted

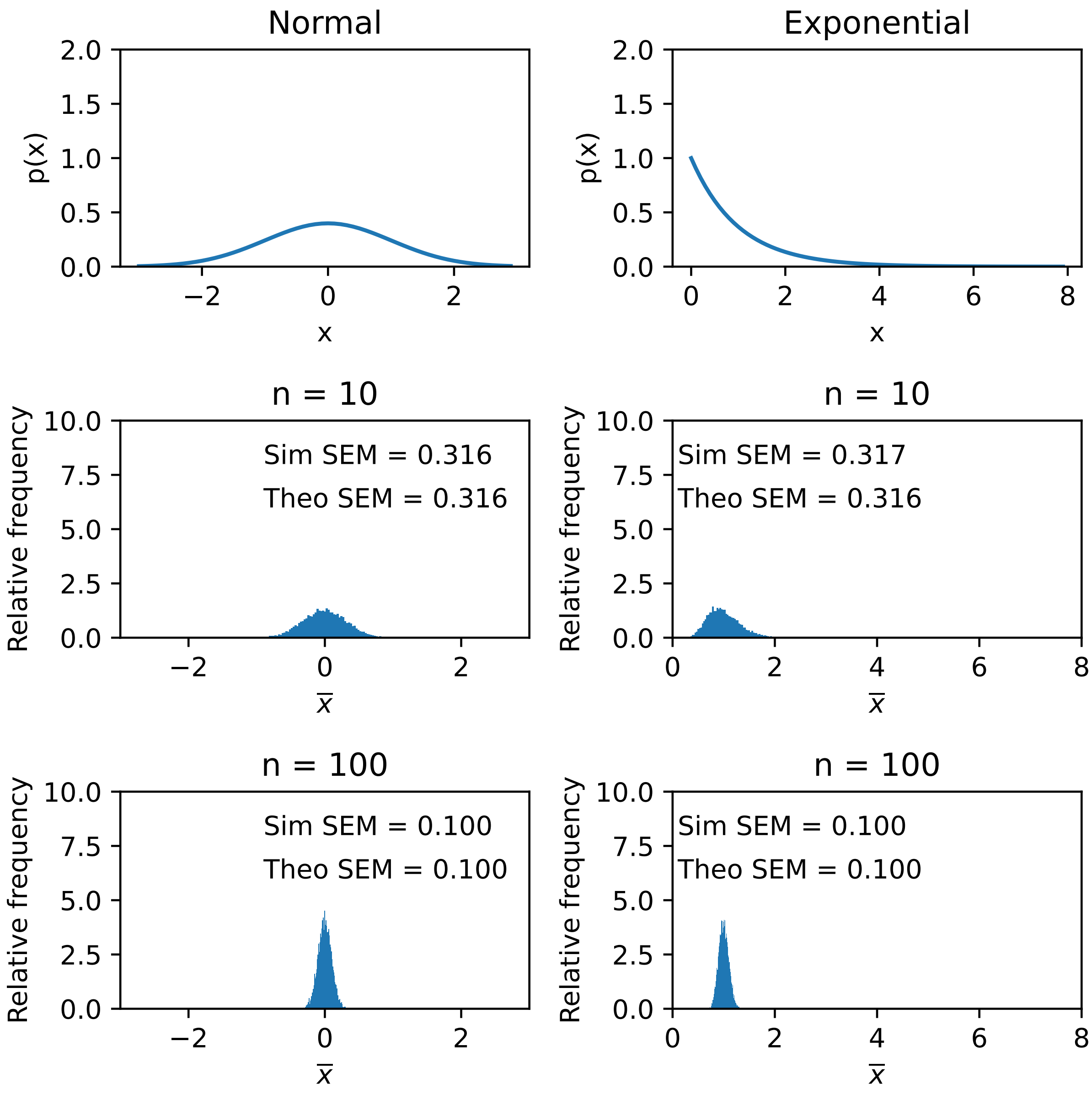


Figure 15.2: Standard error of the mean as the sample size n increases. Top row: normal probability distribution with $\mu = 1$ and $\sigma = 1$ (left) and exponential probability distribution with $\lambda = 1$ (right). Middle row: Histograms showing distribution of the sample mean $\overline{X}$ in 10,000 simulations of sampling 10 random numbers from a normal and exponential distributions. The simulated standard error of the mean (Sim SEM) is calculated from the simulated means. The theoretical standard error of the mean (Theo SEM) is calculated from the formula $\sigma_{\overline{X}} = \sigma/\sqrt{n}$. Bottom row: same as middle row, but with $n = 100$. As n increases, the SEM decreases. Again the simulated SEM is very similar to the estimated SEM. Note that the distribution of the mean of the samples from the exponential distribution for $n = 100$ is almost normal; for $n = 10$ it is somewhat skewed right.

for both a normal distribution (top row) and exponential distribution (bottom row).

Estimated standard error of the mean In the much more realistic case where we don't know the standard deviation of the population, the standard error of the mean is $\hat{\sigma}_{\hat{\mu}} = S/\sqrt{n}$, where S is the sample standard deviation. Since S varies depending on the sample, it's a random variable, and therefore the estimated SEM itself is a random variable.

Figure 15.3 shows the distribution of the SEM in statistical simulations of sampling $n = 10$ or $n = 100$ samples from a normal and exponential distribution. For $n = 10$ it can be seen that the SEM varies a lot around the theoretical value. This means that a particular sample might give us a much higher or lower SEM then the true value. For $n = 100$, the estimated SEM is distributed much more tightly around the theoretical value – it's therefore safer to use the estimated SEM as a substitute for the true SEM.

Related Workshop: Statistical problems 1

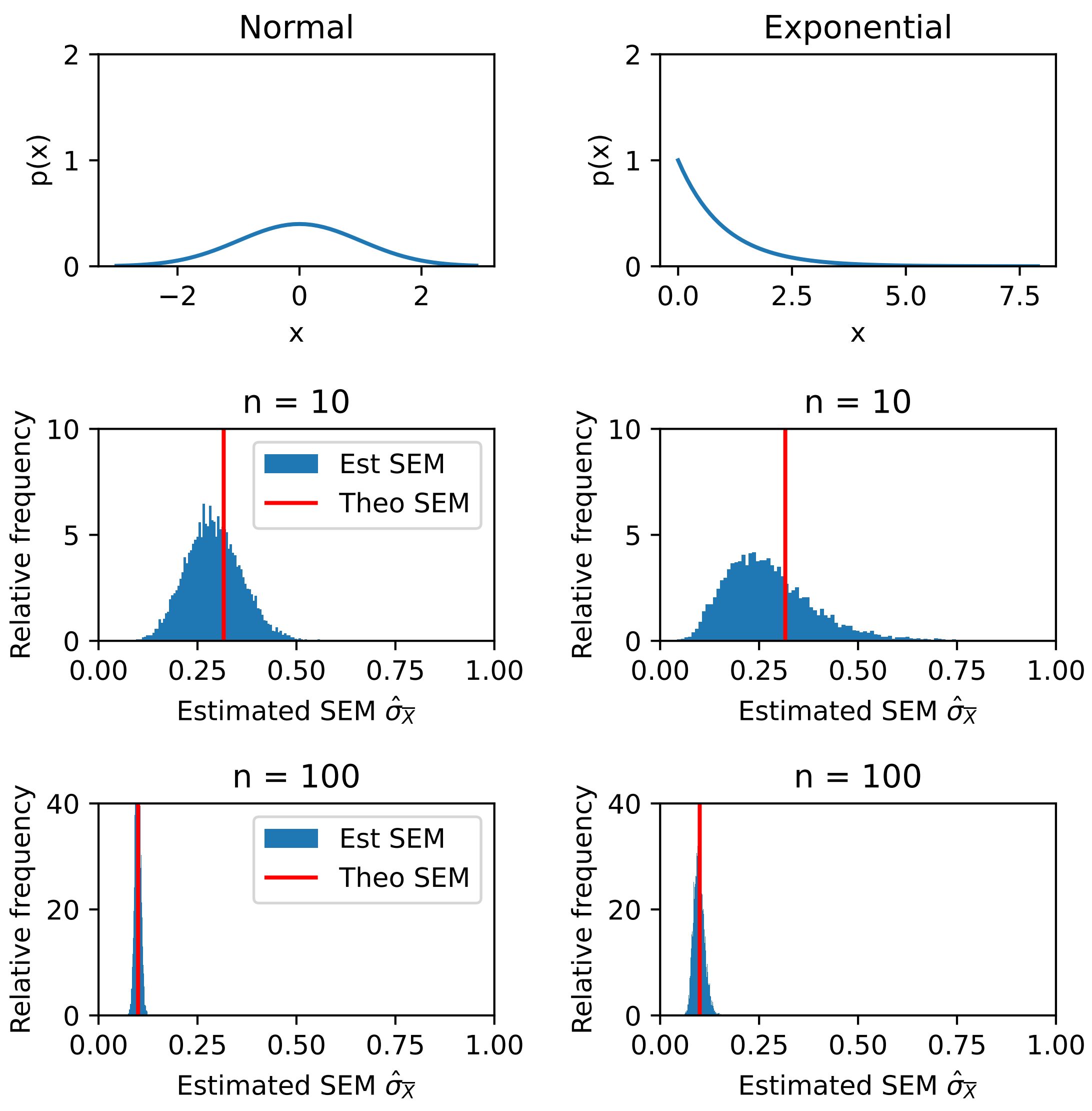


Figure 15.3: Distribution of estimated standard error of the mean. Top row: normal probability distribution with $\mu = 1$ and $\sigma = 1$ (left) and exponential probability distribution with $\lambda = 1$ (right). Middle row: Histograms showing distribution of the SEM calculated by from each of 10,000 simulations of sampling 10 random numbers from a normal and exponential distributions. Bottom row: same as middle row, but with sample size $n = 100$.

Chapter 16

Confidence intervals

Recommended reading

- *Modern Mathematical Statistics with Applications*, Sections 8.1–8.3 and 8.5

16.1 Principle of confidence intervals

Illustration: confidence intervals for the mean From the Central Limit Theorem we know that for large samples, the distribution of the mean is normal, and that the estimated standard error of the mean should be close to the standard error of the mean. We can then ask "if we looked at an interval around our estimate for the mean, how often would the true value be contained in that interval"?

Figure 16.1 gives an illustrated answer to this question. Each blue horizontal line corresponds to one sample of size n from a population, and shows a range of estimates for the population mean based on that sample – in other words a **confidence interval**. We can see that the true value of the mean (black vertical line) is contained in most of the intervals, but not all of them.

Size of confidence interval We have chosen the length of the intervals to ensure that, if we carried on estimating the mean and the interval, about 95% of intervals would contain the true mean. To determine this length, we use the z critical values of the standardised normal distribution with zero mean and variance 1 (Figure 16.2), which we refer to as the ***z*-distribution**. We define the ***z* critical value** z_α as the value of z in a normal distribution which has the area α under the curve to its right. If we want the intervals to contain the true mean 95% of the time, we need to make sure that the mean is within the central 95% of the distribution. This implies that we need 2.5% of the area under the curve to the right of the upper bound, so we look up $z_{0.025}$ in a statistical table or a function in a stats package and find that $z_{0.025} = 1.96$ – we will show how to do this later. The z critical value of 1.96 tells us that the length of the lines on the side of each estimate of the mean should all be 1.96 times the standard error of the mean (SEM).

We may want to be more or less certain of whether the mean is contained in a confidence interval. In this case we can look up the z critical value for our chosen level of confidence. We can also decide to express the confidence interval in terms of the multiples of the SEM. For example confidence intervals of plus or minus one SEM correspond to a 68% confidence interval.

Reminder It is worth remembering that these simulations are artificial in the sense that we can repeat many samples. In real life we only get one sample, which does or does not contain the true value – but we don't know.

Looking up a *z* critical value To look up a z critical value, you can use the python `scipy` package. For example to find $z_{0.2}$ you would use:

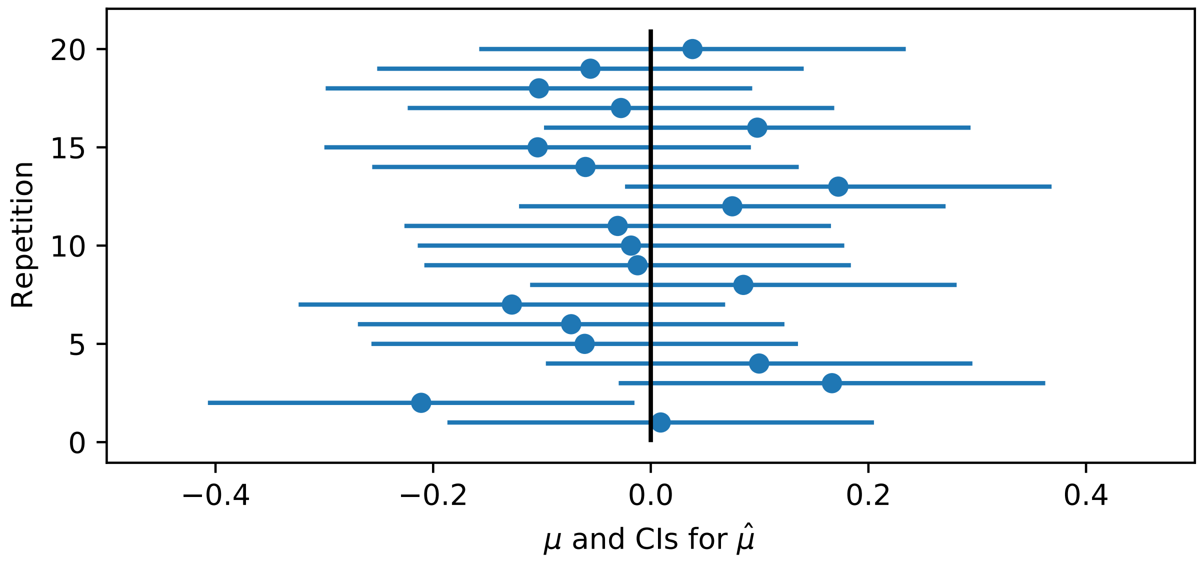


Figure 16.1: Principle of confidence intervals. We repeat a simulation using a sample size of $n = 100$ to estimate the sample mean of a normal distribution with mean 0 and standard deviation 1. The black vertical line indicates the true mean, the blue dots indicate the sample means, and the blue horizontal lines indicate the 95% confidence intervals obtained in each of the 20 repetitions. It can be seen that 19 of the confidence intervals do contain the population mean, but one of them does not.

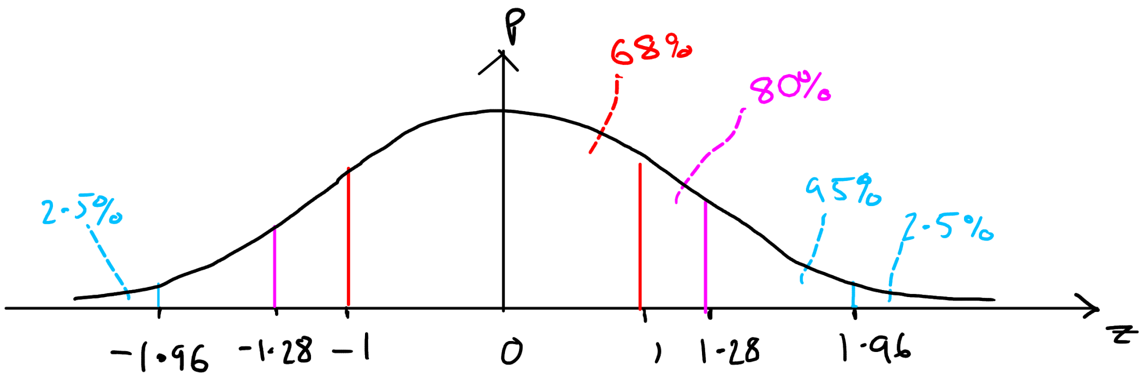


Figure 16.2: Confidence intervals of a normal distribution. The intervals containing various amounts of probability mass under a normal distribution are shown. The 95% confidence interval (blue) is $[-1.96, 1.96]$ and has 2.5% of the probability mass in each tail. The 80% confidence interval is $[-1.28, 1.28]$. The amount of probability mass contained in one standard deviation is 68%. In general for a confidence interval of $100(1-\alpha)\%$, the upper and lower boundaries are determined by the z critical value $z_{\alpha/2}$. E.g. with the 95% confidence interval $\alpha = 0.05$ and there is 2.5% of the area of the curve above the upper boundary of the confidence interval.

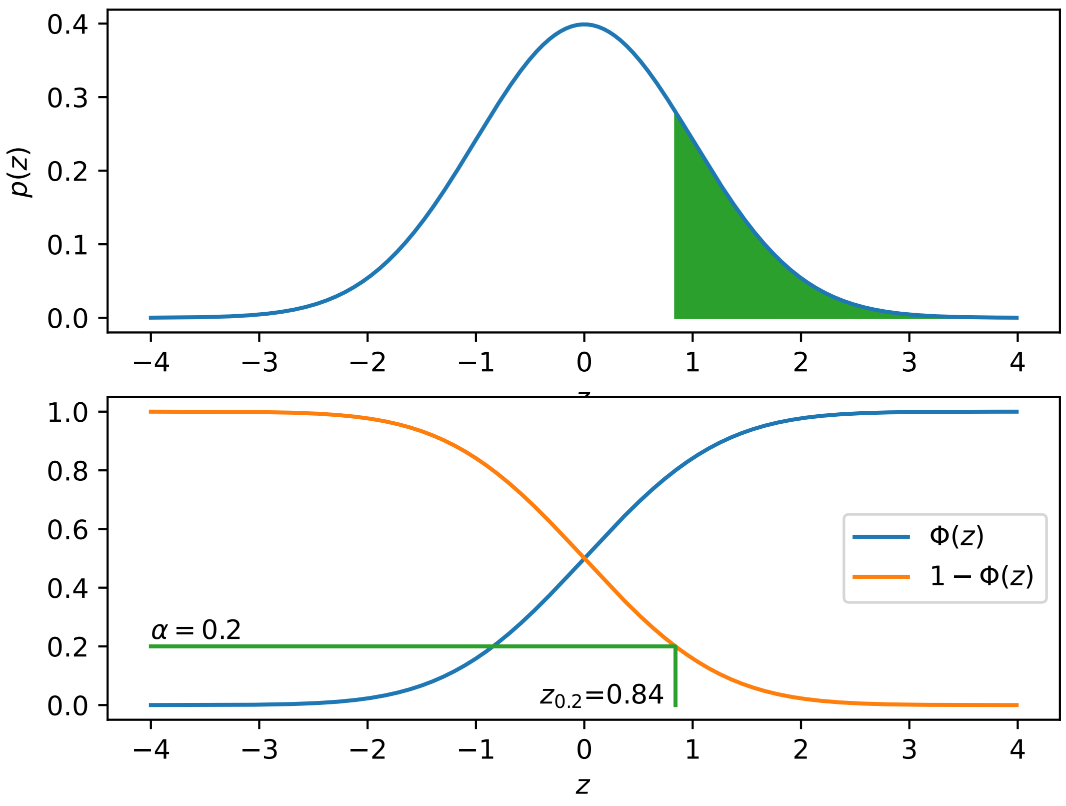


Figure 16.3: Concept of the z critical value. Top: z_α is the value of z such in a normal distribution such that the area under the curve to the right of z_α (green) is equal to α. i.e. $\alpha = \int_{z_\alpha}^{\infty} p(z)dz$. Bottom: the blue curve shows the cumulative distribution function $\Phi(z) = \int_{-\infty}^{z} p(z)dz$. The orange curve shows the "survival function" $\mathrm{sf}(z) = 1 - \Phi(z)$. The survival function of z is exactly the area to the right of z under the pdf. Therefore we want to look up the inverse survival function to determine z_α from α, as indicated by the green lines.

Table 16.1: Abbreviated table of critical values for t and z distributions.

α / ν	0.100	0.050	0.025	0.010	0.005	0.001
1	3.078	6.314	12.706	31.821	63.657	318.309
2	1.886	2.920	4.303	6.965	9.925	22.327
3	1.638	2.353	3.182	4.541	5.841	10.215
4	1.533	2.132	2.776	3.747	4.604	7.173
5	1.476	2.015	2.571	3.365	4.032	5.893
6	1.440	1.943	2.447	3.143	3.707	5.208
7	1.415	1.895	2.365	2.998	3.499	4.785
8	1.397	1.860	2.306	2.896	3.355	4.501
9	1.383	1.833	2.262	2.821	3.250	4.297
10	1.372	1.812	2.228	2.764	3.169	4.144
20	1.325	1.725	2.086	2.528	2.845	3.552
30	1.310	1.697	2.042	2.457	2.750	3.385
40	1.303	1.684	2.021	2.423	2.704	3.307
∞	1.282	1.645	1.960	2.326	2.576	3.090

```
from scipy.stats import norm
alpha = 0.2
print(norm.isf(alpha))
```

The function name `isf` stands for **inverse survival function**. As illustrated in Figure 16.3, it's the inverse of one minus the cumulative distribution function (cdf).

You can also look up z critical values in statistical tables, such as the ones in the appendices of *Modern Mathematical Statistics with Applications*. Table 16.1 shows an abbreviated example of such a table. The final row (lablelled ∞, for reasons to be explained later on) contains the z critical values for the values of α shown in the table header. The meaning of the rows will be explained later (see Confidence intervals on the mean for small samples).

16.2 Definition of confidence intervals

Definition of confidence intervals We define a **confidence interval** as an interval $(\hat{\vartheta} - a\hat{\sigma}_{\hat{\vartheta}}, \hat{\vartheta} + b\hat{\sigma}_{\hat{\vartheta}})$ that has a specified chance $1 - \alpha$ of containing the parameter, and where the positive numbers a and b defining the lower and upper bounds of the interval depend on α. The smaller α is, the larger the values of a and b can be for the statement to hold. A common value for α is 0.05 (i.e. 5%), which gives a 95% confidence interval. However, we could set $\alpha = 0.2$, which would give a narrower 80% confidence interval. Often a and b are equal, but we have given them distinct symbols for full generality.

We can express the definition in terms of a probability statement as follows:

$$P\left(\hat{\vartheta} - a\hat{\sigma}_{\hat{\vartheta}} < \vartheta < \hat{\vartheta} + b\hat{\sigma}_{\hat{\vartheta}}\right) = 1 - \alpha \tag{16.1}$$

In this probability statement, the upper and lower bounds of the interval are random variables, since they are based on the estimators and the estimated standard error, which are themselves random variables derived from the sample.

Expression in terms of random variable in fixed interval We can rearrange the definition of the confidence interval in terms of a standardised variable $(\hat{\vartheta} - \vartheta)/\hat{\sigma}_{\hat{\vartheta}}$:

$$P\left(-b < \frac{\hat{\vartheta} - \vartheta}{\hat{\sigma}_{\hat{\vartheta}}} < a\right) = 1 - \alpha \tag{16.2}$$

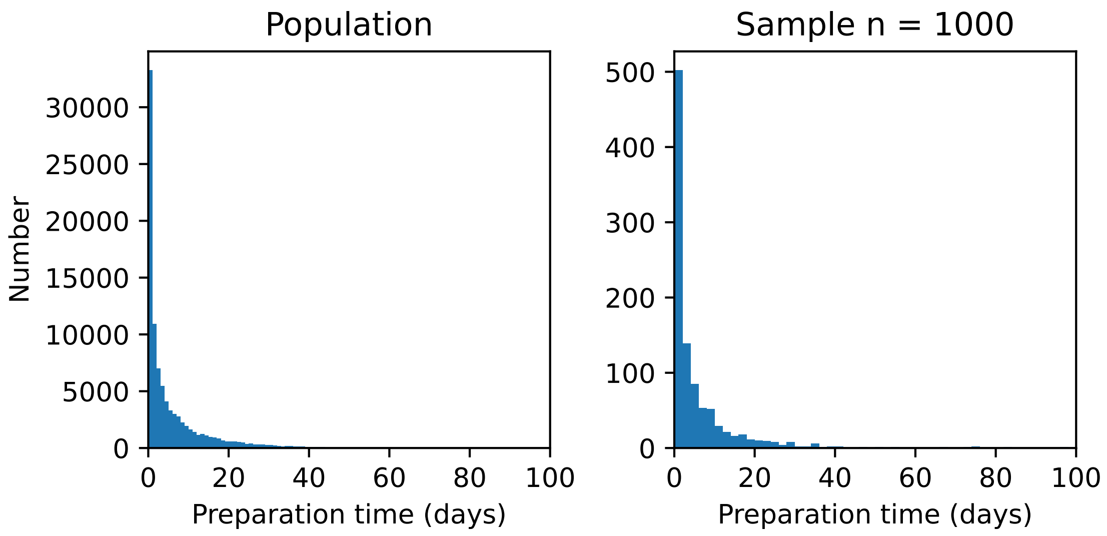


Figure 16.4: Distribution of time from making a reservation to the reservation time ("preparation time") in restaurants using the "air" booking system in Japan in the period January 2016–April 2017.

Because this standardised variable is derived from the sample, it fits our definition of a statistic. Furthermore, it is composed of *two* statistics, the estimator $\hat{\vartheta}$ and the estimated standard error $\hat{\sigma}_{\hat{\vartheta}}$.

16.3 Method of estimating confidence interval of the mean of a large sample

Methods of estimating confidence intervals There are two main methods of estimating confidence intervals:

1. Under some assumptions about the distribution of the data X_i and the number of samples n we can derive the distribution of $(\hat{\vartheta} - \vartheta)/\hat{\sigma}_{\hat{\vartheta}}$, which will then tell us the values of $-b$ and a at the $100\alpha/2$th centile and the $100(1-\alpha/2)$th centiles.

2. More generally we can use a type of statistical simulation called a bootstrap estimator to derive the confidence interval.

We'll demonstrate the first approach by continuing with our simplified example of a normal distribution with known parameters. In the following section we'll then cover the bootstrap estimator.

Example: confidence interval for the mean of a normal distribution with known variance In the example of sampling from a normal distribution introduced in the last section, we *know* the population variance σ, and by definition, the standard estimate of the mean is $\hat{\sigma}_{\hat{\vartheta}} = \sigma/\sqrt{n}$. Because the population variance σ is known, it's not a random variable, and therefore the SEM $\hat{\sigma}_{\hat{\vartheta}}$ isn't a random variable either. The standardised variable in Equation 16.2 is therefore

$$\frac{\hat{\vartheta} - \vartheta}{\hat{\sigma}_{\hat{\vartheta}}} = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \tag{16.3}$$

and only contains one random variable, $\overline{X}$. This makes it quite easy to deal with, since we know that this distribution is a standard normal distribution, so we can define a 95% confidence interval by setting a and b to be values at which the cumulative distribution function (cdf) is equal to 2.5%($= \alpha/2$) and 97.5%($= 1-\alpha/2$). In this case the values $a = b = 1.96$ satisfy these conditions. Generally we set a and b symmetrically, so that there is equal weight in the "tails" of the distribution (Figure 16.2).

Table 16.2: Summary statistics of population and sample of preparation times, generated by the pandas `describe` function.

	Population	Sample
count	92378.00	1000.00
mean	8.30	8.06
std	25.65	27.72
min	0.00	0.00
25%	0.21	0.17
50%	2.08	1.96
75%	7.88	6.92
max	393.12	364.96

Confidence intervals for the mean of a large sample The central limit theorem states that the distribution of the sample mean of a "large" sample from any distribution should be normal. How large the sample needs to be depends on the distribution, but Figure 16.1 demonstrates that sample means of $n = 100$ samples from an exponential distribution already appear to fairly normally distributed with the SEM as predicted to be the standard deviation of the exponential distribution divided by $\sqrt{n}$. This means that we can use the procedure above to find confidence intervals.

Confidence intervals for the mean of an empirical distribution Up until now, we have considered estimating parameters from theoretical probability distributions, such as the normal distribution or the exponential distribution. We'll now consider how we can estimate the parameters of an **empirical distribution**, i.e. real-world data, from a sample of that distribution.

As an example, we will take the population of times between making a reservation and the time of the reservation itself in Japanese restaurants using the "air" booking system. The full population contains 92378 times (Figure 16.4, left) and we've created a random sample of 1000 of these times (Figure 16.4, right). In real life, if we had the full set of data, there would not be any point in creating this random sample of times, but we do so here to demonstrate how well we can estimate confidence intervals. From now on imagine that the sample of 1000 times is all that we have available to us. It's important to notice that the distribution of the sample resembles the population distribution, even though it is rougher.

Table 16.2 shows the summary statistics for the population and the sample. We can see that the estimates for the mean, standard deviation and centiles from the sample are all similar to the true population values. From the table we can see that the population mean is $\mu = 8.30$ days and the sample mean is $\bar{x} = 8.06$ days. The sample mean would be different if we'd happened to have taken a random different sample.

From the summary statistics from the sample, we have $\bar{x} = 8.06$ days and the standard deviation $s = 27.72$ days. Our estimator for the mean is $\hat{\vartheta} = \bar{x} = 8.06$ days. Our estimator for the standard error of the mean is $\hat{\sigma}_{\hat{\vartheta}} = s/\sqrt{n} = 27.72/\sqrt{1000} = 0.88$ days. The 95% confidence interval for the mean in days is therefore $(\hat{\vartheta} - 1.96\hat{\sigma}_{\hat{\vartheta}}, \hat{\vartheta} + 1.96\hat{\sigma}_{\hat{\vartheta}}) = (6.34, 9.78)$.

Reporting confidence intervals When reading scientific papers, there are various ways of reporting confidence intervals:

- M=8.06, CI=6.34–9.78. Here "M" stands for mean and "CI" stands for confidence interval.
- 8.06 ± 1.72 (95% confidence interval)
- 8.06 ± 0.88 ($\pm$ 1 SEM). This is a 68% confidence interval, though the confidence interval isn't specified in terms of area under the curve.
- 8.06 ± 1.76 ($\pm$ 2 SEM).

Figure 16.5: Bootstrapping: Baron Münchhausen pulls himself and his horse out of a swamp by his pigtail. Public domain image from Wikipedia's article on bootstrapping.

16.4 Bootstrap estimation of confidence intervals

Principle of a bootstrap estimator We want to estimate the standard error of an estimator. In the topic on sampling, we have already seen what happens when we sample a mean repeatedly from a theoretical distribution, and that this can give us a measure of the standard error of the estimator.

The name "bootstrap estimator" arises because it appears to do something physically impossible, such as "pulling ourselves up by our own bootstraps". (Equivalently we could pull ourselves up by our pigtail, Figure 16.5).

In a bootstrap estimator we treat the sample that we have available as a population, and resample from it to give the sampling distribution of the estimator. From the sampling distribution we can compute the standard error of the estimator. It feels as though we shouldn't be able to treat the sample as a population, but it works because if we have a large enough sample, the distribution of the sample will resemble the population itself.

Bootstrap procedure for finding a confidence interval for the mean We will start with a large sample n from the data, which has a mean $\overline{x}$. By large, we mean large enough that the sample resembles the population distribution. Of course, this is not possible to know exactly, so the larger the better. We decide to take B bootstrap samples. Common numbers are 1000 or 5000, or 10000. More samples are generally better, but bootstrapping can be computationally expensive, and fewer samples can also give reasonable results.

Here is the procedure:

- For j in $1, \dots, B$
  - Take sample x^* of size n from the sample *with replacement*
  - Compute the sample mean of the new sample $\overline{x_j^*}$
- To compute the bootstrap confidence interval, we find the centiles of the distribution at $100\alpha/2$ and $100(1-\alpha/2)$. We can do this by arranging the sample means $\overline{x_j^*}$ in order from lowest to highest, and pick $\overline{x_j^*}$ at $k = \alpha(B+1)/2$ to be the lower end of the CI and pick $\overline{x_j^*}$ at $k = B - \alpha(B+1)/2$ to be the upper end of the CI.

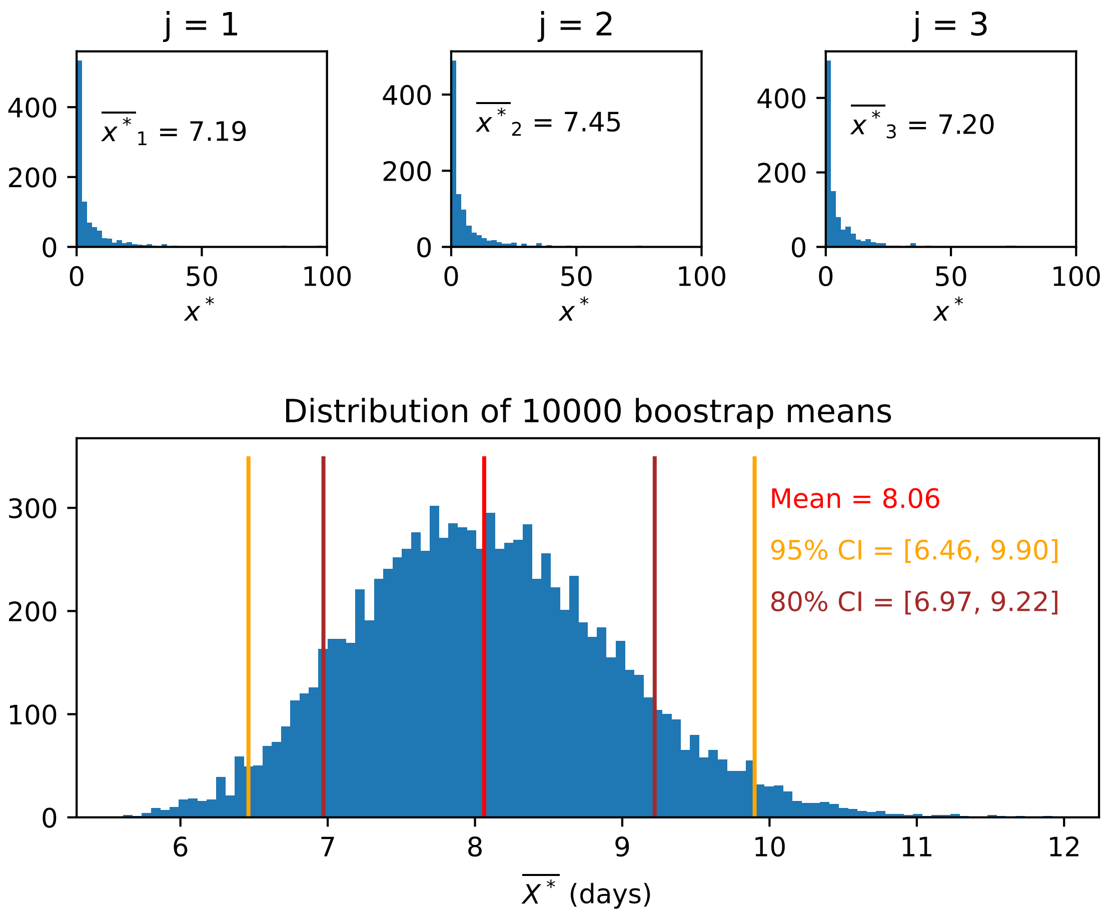


Figure 16.6: Demonstration of bootstrap mean applied to restaurant reservation time data (Figure 16.4). The top row shows the distributions obtained from the first 3 of 10000 bootstrap samples. Although the distributions are similar to each other, they are not exactly the same, and the sample mean of each is different. The bottom figure is the distribution of all 10000 of these bootstrap sample means. The mean of the original sample is shown, as is the 95% and 80% confidence intervals.

- We can also compute the bootstrap estimator of the variance of the mean:

$$s^2_{\text{boot}} = \frac{\sum_{j=1}^{B}(\overline{x^*_j} - \overline{x})^2}{B-1}$$

The advantages of the bootstrap procedure are that we can use it for any estimator, e.g. the median, and that we do not need to make any assumptions about the distribution of the estimator.

Example of bootstrap estimator applied to the mean We'll now apply the bootstrap estimator to give us a confidence interval for the mean (Figure 16.6). For each of our 10,000 bootstrap samples, we'll resample 1000 samples *with replacement* from our sample of 1000. Each of these samples will be a distribution (top row of Figure 16.6), from which we can compute the 10,000 bootstrap means. Then we'll plot the distribution of the bootstrap means (bottom row of Figure 16.6) and find the 95% and 80% confidence intervals. In this case we can see both the 95% and 80% confidence intervals contain the *population* mean (8.30, Table 16.2). However, if we replicate the experiment with a different initial random sample of 1000, in around 5% of cases we should expect that the 95% confidence interval does not contain the mean.

We'll leave this as a lab exercise for you to implement, though you will find that you get different answers for the confidence intervals, depending on the state of the random number generator.

Comparison of bootstrap confidence intervals with normal approximation The 95% confidence interval obtained via the bootstrap procedure is (6.46, 9.90) days, which is very similar to the confidence interval obtained by the normal approximation, (6.34, 9.78) days. The bootstrap interval is slightly shifted to the right, suggesting that the normal approximation is quite accurate at a sample size of $n = 1000$.

General formulation of bootstrap estimator A great advantage of the bootstrap is that we can easily apply it to statistics other than the mean. Here is the general procedure for estimating the confidence interval of a generic estimator $\hat{\vartheta}$:

- For j in $1 \dots B$
  - Take sample x^* of size n from the sample *with replacement*
  - Compute the sample statistic of the new sample $\hat{\vartheta}^*_j$
- Then compute the bootstrap estimator of the variance of the statistic:

$$s^2_{\text{boot}} = \frac{\sum_{j=1}^{B}(\hat{\vartheta}^*_j - \hat{\vartheta})^2}{B-1}$$

- To compute the bootstrap confidence interval, we find the centiles of the distribution at $100\alpha/2$ and $100(1-\alpha/2)$.

This procedure works well for measures of centrality such as the median, and for the variance. It doesn't work so well for statistics of extremes of the distribution, such as the maximum or minimum.

16.5 Interpretation of confidence intervals

Interpretation of confidence intervals Although we have only computed confidence intervals in a simple artificial example, we are already at a stage where we can consider how to interpret confidence intervals. From Equation 16.1 we can see that confidence intervals are a random interval – whenever we take a new sample, we will end up with a new interval, as illustrated in Figure 16.1. The interpretation (according to the frequentist interpretation of statistics) is that if we performed a long run of experiments (i.e. repeatedly took samples) the parameter (the mean in this case) would be in around 95% of the confidence intervals.

How big should a confidence interval be? Should we choose the 95% confidence interval or the 80% confidence interval? The answer to this question depends on the problem. For example, suppose we have a machine that makes tens of thousands of ball bearings for aircraft jet engines every day. Each ball bearing needs to have a diameter of 2 ± 0.0001mm for the engine to work safely. We measure the diameter of a sample of the ball bearings every day. Because this is a safety-critical application, we need to have high confidence (say 99.999%) that the ball bearings are in the range 2 ± 0.0001mm. This might require a large sample size, but it's worthwhile because the consequences of getting it wrong could be catastrophic.

On the other hand, suppose we are estimating the number of red squirrels in a population so that we know how much red-squirrel friendly food to put out for them over winter. We might want to leave out a bit more than we expect they need, we're happy to accept a 10% chance that the true number of squirrels might be greater than the upper end of a confidence interval, so we compute the 80% confidence interval, and put out enough food for the number of squirrels at the upper end of the interval. There's a 10% chance that we might not be providing for enough squirrels, but it's not as catastrophic as in the aircraft situation (depending on how much you value red squirrels compared to humans).

Upper and lower confidence bounds In this case, we're not worried about our estimate being too low, so we only need to compute the upper confidence bound – we would quote a mean number of squirrels and an upper limit.

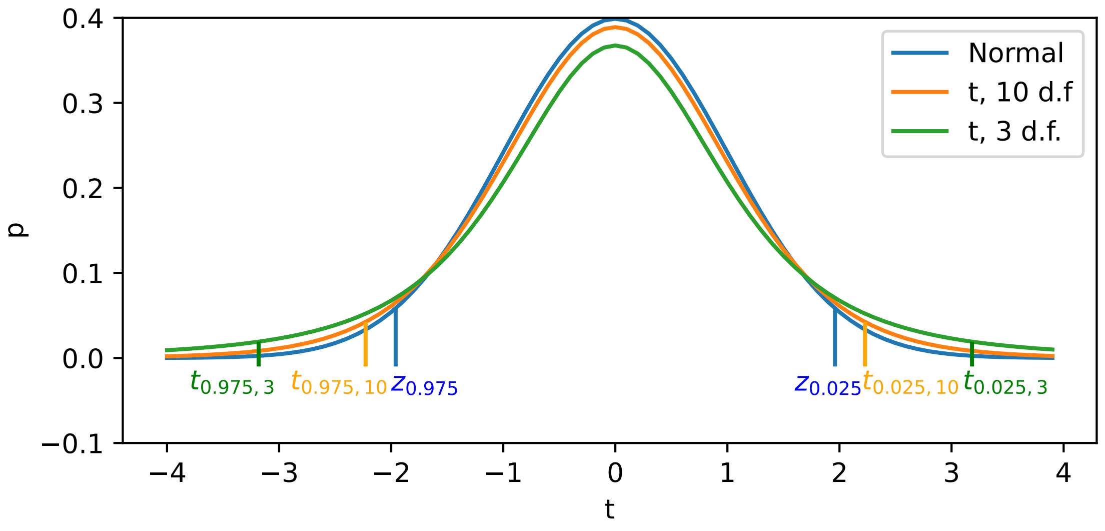


Figure 16.7: The t-distribution for 3 degrees of freedom and 10 degrees of freedom, with normal distribution for comparison. 2.5% t critical values and z critical values are shown.

16.6 Confidence intervals on the mean for small samples

Small samples We'll now consider the distribution of the mean based on a sample of a "small" number (usually $n < 40$) of data that appears to be distributed normally and whose variance we are not given – we can only estimate it from the data.

For example, suppose we want to estimate the mean weight of a population of female squirrels from a sample of $n = 32$ squirrels (Wauters and Dhondt, 1989). The sample mean is $\overline{x} = 341.0$g and the estimated standard error of the mean is $\hat{\sigma}_{\overline{X}} = 3.9$g.

We can imagine that if we had taken a different sample of $n = 32$ squirrels, we would have found both a different sample mean and a different estimate for the standard error of the mean. Thus, the standardised statistic $(\overline{X} - \mu)/\hat{\sigma}_{\overline{X}}$ itself contains *two* statistics, $\overline{X}$ and $\hat{\sigma}_{\overline{X}}$, which are, in general, random variables derived from the sample. As we are estimating the standard deviation, rather than knowing it, the normal approximation to the distribution of the mean begins to break down.

The *t*-distribution We could use the bootstrap estimator to estimate confidence intervals. However, in this special case, there is another option. There's a theorem that states that when X is a random sample of size n from a normal distribution with mean μ, the random variable

$$T = \frac{\overline{X} - \mu}{\hat{\sigma}_{\overline{X}}}$$

is distributed as a ***t*-distribution** with $n - 1$ degrees of freedom, where the t-distribution with ν degrees of freedom has the probability density function depicted in Figure 16.7, which is given by the equation:

$$p_\nu(t) = \frac{1}{\sqrt{\pi\nu}} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)} \frac{1}{(1 + t^2/\nu)^{(\nu+1)/2}}$$

where $\Gamma(x)$ is a gamma function. We will not prove this theorem here; in *Modern Mathematical Statistics with Applications* Section 6.4 there is the sketch of a proof.

The t-distribution is very similar in shape to the normal distribution: it is bell-shaped, symmetrical, and centred on 0. However, for small numbers of degrees of freedom, has longer tails. This means that the tails contain more of the weight of the distribution. We define the ***t* critical value** $t_{\alpha,\nu}$ as the value of t in a t-distribution with ν degrees of freedom which has the area α under the curve to its right.

For small degrees of freedom, the t critical values are considerably bigger than the z critical values of the normal distribution (Figure 16.7). As the number of degrees of freedom increases, the t-distribution approaches a normal distribution. The distribution with 40 degrees of freedom (not shown in the figure) looks very similar to a normal distribution.

Looking up a t critical value To look up a t critical value, you can use the python `scipy` package. For example to find $t_{0.025,10}$ you would use:

```
from scipy.stats import t
alpha = 0.025
nu = 10
t_cv = t(nu).isf(alpha)
print(t_cv)
```

You can also look up t critical values and z critical values in statistical tables, such as the ones in the appendices of *Modern Mathematical Statistics with Applications*. Table 16.1 shows an abbreviated example of such a table. Each row contains t critical values for degree various levels of α. The final row, with infinite number of degrees of freedom, is the z critical values for these values of α. The full tables include values for more degrees of freedom.

Using the t-distribution to derive a confidence interval The $100(1-\alpha)$ percent confidence interval around a mean $\overline{x}$ of a sample of n values with estimated SEM $\hat{\sigma}_{\overline{X}}$ derived using a t-distribution is:

$$(\overline{x} - t_{\alpha/2,n-1}\hat{\sigma}_{\overline{X}}\ ,\ \overline{x} + t_{\alpha/2,n-1}\hat{\sigma}_{\overline{X}}) \tag{16.4}$$

Note that we have used the t critical value $t_{\alpha/2,\nu}$. Here the number of degrees of freedom is one less than the sample size ($\nu = n-1$). Also, we have divided α by 2 because we are wanting upper and lower bounds to the confidence interval. It might be that we only need an upper bound, as we considered when we were estimating squirrel numbers earlier. In this case we would just quote $\overline{x} + t_{\alpha,n-1}\hat{\sigma}_{\overline{X}}$. This is still a $100(1-\alpha)$ confidence interval, since the interval from $-\infty$ to the upper bound contains $100(1-\alpha)$ of the area under the t-distribution.

To continue the squirrel example, suppose we want to find a 95% confidence interval for the weight. The 95% confidence interval implies $\alpha = 0.05$ and $\nu = n-1 = 31$. We would then look up the $t_{0.025,31} = 2.040$ and substitute it into Equation 16.4 along with the sample mean and estimated SEM, and then use this to generate the confidence interval, which we could quote as $\hat{\mu} = 341.0 \pm 8.0$g (95% confidence interval, $n = 32$). This is a bit wider than the interval we would obtain using the corresponding critical value of a normal distribution $z_{0.025} = 1.96$.

Related Python Lab: Estimation of confidence intervals with the bootstrap

`https://github.com/Inf2-FDS/FDS-S1-11-estimation-bootstrap`

Related Workshop: Statistical problems 1

Chapter 17

Hypothesis testing and *p*-values

Recommended reading

- XKCD comic strip on multiple testing – funny!
- *A hypothesis is a liability* Yanai and Lercher (2020) – thought-provoking and amusing article

17.1 Principle of hypothesis testing

Hypothesis testing helps us to answer yes/no questions, such as "is chocolate good for you?" or "is a jury selection procedure biased?" There are two aspects to hypothesis testing:

1. Deciding on whether a **hypothesis** or **model** is compatible with data from observational studies and randomised experiments.
2. If the hypothesis is compatible with the data, investigating the mechanisms specific to the data, e.g. the biological effect of chocolate on the body or the process by which a jury panel was selected.

In the course we are going to focus on the statistical aspects (Aspect 1), but it's worth remembering that the question is not answered once we've completed this step – it should prompt further investigation of the question rather than ending the inquiry (Yanai and Lercher, 2020). Furthermore, as Yanai and Lercher (2020) illustrate rather amusingly, it is important to explain data before undertaking hypothesis testing – a good visualisation can reveal features of the data that a hypothesis test can't.

Method of hypothesis testing At the core of hypothesis testing are the **null hypothesis** and the **alternative hypothesis**:

The null hypothesis H_0**:** The claim that we initially assume to be true, formalised as a statistical model. e.g. "The jury panel was chosen by random selection from the population in a district."

The alternative hypothesis H_a**:** The claim that is contradictory to H_0, typically not formalised as a statistical model. E.g. "The jury panel was chosen by some other, unspecified, method."

The aim of hypothesis testing is to either reject or not reject the null hypothesis. Note that we do not "accept" the null hypothesis as true, we are just saying that it's not been proved to be false.

Test procedure The procedure to carry out a hypothesis test, which we call the **test procedure**, consists of:

1. Deciding on a **test statistic**, which is a function of the sample data, e.g. the number of Black people in a jury panel.

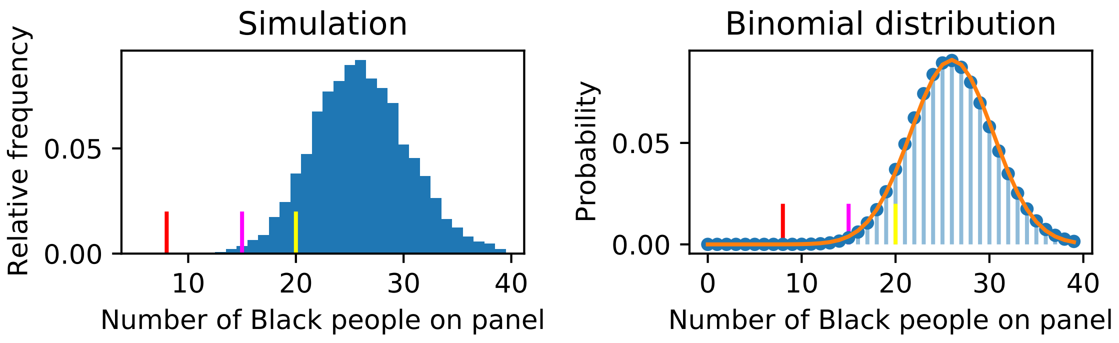


Figure 17.1: Distributions of number of Black people T_0 (test statistic) on a panel of 100 under the null hypothesis that the jury was randomly selected from a population that is 26% Black and 74% non-Black. Left: distribution arising from 10 000 statistical simulations. The red line indicates the number of Black jurors in Swain versus Alabama (1965), the magenta line indicates $t_0 = 15$ and the yellow line indicates $t_0 = 20$. Right: Binomial distribution (blue dots) for $n = 100$ and $p = 10$. Normal approximation (orange curve) with $\mu = np$ and $\sigma = \sqrt{np(1-p)}$.

2. Determining what the distribution of the test statistic would be if it arose from the null hypothesis statistical model.

3. Either:

   (a) Deciding on a **rejection region**, i.e. regions of the distribution of the test statistic under H_0 in which we should reject H_0. Typically, these are the extremities of the distribution. If our test statistic falls into the rejection region, we reject H_0; otherwise, we don't reject it.

   (b) Returning a ***p*-value**, which tells us how compatible the test statistic is with the distribution predicted by chance from H_0.

Application of test procedure to example In the topic on Randomness, sampling and simulation, we looked at the example of Swain versus Alabama (1965), in which the question was "if 8 Black people were chosen for a jury panel of 100 people, but the fraction of Black people in the population was 26%, does this show bias against Black people?" We found the distribution of the test statistic under the null hypothesis by simulating the null hypothesis model of sampling from a Bernoulli distribution with $P(\text{Black}) = 0.26$. In this case probability theory also tells us that the distribution is a binomial distribution with $n = 100$ and $p = 0.26$. We found that there were no replications in which 8 Black members were chosen (Figure 17.1) – the simulated numbers were always higher.

We did not consider rejection regions or p-values. Since the observed data (8 Black people on the panel; red line in Figure 17.1) were inconsistent with the range of predictions produced by the null hypothesis, it seemed very clear that we should reject the null hypothesis. But what would we have decided if the number of Black people had sat within the distribution of simulated values, e.g. 15 (magenta line) or 20 (yellow line)?

Rejection regions We might want to specify the rejection region as the bottom 5% of the probability mass, i.e. the region that seems unusually low (Figure 17.2, left, region to left of orange boundary). If the observed test statistic falls into that region, we might "reject the hypothesis at the 5% level (one-tailed test)". We call this a **one-tailed test** because the rejection region occupies only one tail of the distribution. This is justified, as the alternative hypothesis was implicitly "the number of Black people selected is below the number we would have expected by chance".

If we know the distribution of our null hypothesis model, we can look up statistical tables to determine the boundaries of rejection regions. E.g. in this case, the number n is large enough that we can approximate the binomial distribution with a normal distribution with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$. This means

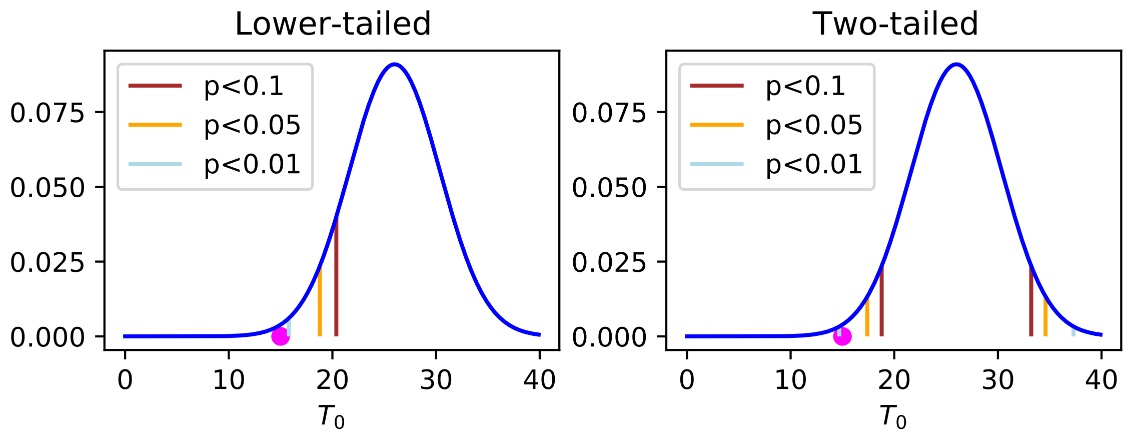


Figure 17.2: Rejection regions. Lower-tailed (left) and upper-tailed (right) rejection regions are shown for the normal approximation to the distribution of the null hypothesis model in the Swain–Alabama example. The observed statistic $t_0 = 15$ is shown with a magenta dot. It lies in the $p < 0.01$ rejection region for a lower-tailed test and in the $p < 0.05$ rejection region for a two-tailed test.

that the standardised statistic

$$Z = \frac{T_0 - \mu}{\sigma} \tag{17.1}$$

is normally distributed. At the edge of the rejection region, this statistic is equal to the z critical value $z_{0.95}$, which has 95% of the probability mass to its right. We can then rearrange Equation 17.1 to find the edge of the rejection region in terms of the original statistic:

$$T_0 = \mu + \sigma z_{0.95} \tag{17.2}$$

If a test statistic in a hypothesis test is distributed according to a normal distribution, the hypothesis test is sometimes referred to as a "z-test".

One-talied and two-tailed tests We could have formulated the alternative hypothesis as "the number of Black people selected is different from (i.e. above or below) the number we would have expected by chance". In this case we would perform a **two-tailed test** (Figure 17.2, right) by setting the rejection regions to be the bottom 2.5% and the top 2.5% of the probability mass of the distribution. We would "reject the hypothesis at the 5% level (two-tailed test)".

17.2 *p*-values

Principle of *p*-values The principle of p-values is that we set the boundary of the rejection regions to be where the data is, and then report the probability mass in the resulting rejection regions as the ***p*-value**.

Determining *p*-values from statistical simulations Had there been 15 Black people on the panel in Swain versus Alabama (magenta line), a fraction 0.0062 of the 10 000 simulations produced panels with 15 or fewer black members. This would therefore give the p-value $p = 0.0062$, i.e. 0.62%. This certainly calls into question if the observed data is compatible with the null hypotheses.

Suppose that there had been 20 Black people on the jury panel (yellow line in Figure 17.1). The corresponding rejection region is 20 or fewer Black people on the jury. A fraction of 0.101 of the simulations are in this region, so the p-value is $p = 0.101$. We would tend not to reject the null hypothesis at this size of p-value, but this would not mean that the null hypothesis was true.

Table 17.1: P-values computed by various methods for various observed values of t_0 in Swain versus Alabama (1965).

t_0	Simulation	Binomial	Normal
8	0	4.73e-06	2.03e-05
15	0.0067	0.0061	0.0061
20	0.1020	0.1030	0.0857

Sometimes the p-value is reported relative to a round figure rejection region, e.g. in the case with 15 Black people on the jury, $p < 0.01$, indicating that we could "reject the null hypothesis at the 1% level". However, supplying the actual p-value gives more information than just reporting the rejection region.

Determining p-values from probability distributions Sometimes it is straightforward to compute the probability distribution implied by the null hypothesis. In the Swain versus Alabama example, it is a binomial distribution with $n = 100$ and $p = 0.26$ (Figure 17.1, right). As we are looking at a lower-tailed test, the p-value is the cumulative distribution function of the binomial distribution, cut off at t_0, the observed number of Black people on the jury panel:

$$P(T_0 \leq t_0) = B(t_0; n, p) = \sum_{t=0}^{t_0} b(t; n, p) \tag{17.3}$$

where $b(t; n, p)$ is the probability of t successes in a binomial distribution with n trials and success probability p; $B(t; n, p)$ is the corresponding cumulative distribution function (cdf). Stats packages have functions to compute the cdf for various distributions, and the values for the binomial are shown in Table 17.1 along with the simulated values.

Also shown is the normal approximation to the binomial, in which we set $\mu = np$ and $\sigma = \sqrt{np(1-p)}$. The p-values are the values of the normal cumulative distribution function at the standardised value

$$z = \frac{t_0 - \mu}{\sigma} \tag{17.4}$$

Why use rejection regions? The rejection region method works well with printed statistical tables, in which critical values of z and other distributions are available only for particular cut-off values, e.g. 0.01, 0.05. With computer packages it is now possible to define the rejection region relative to the observed data rather than a pre-set cut-off.

Definition of p-value We can define the p-value as follows:

> The p-value is the probability, calculated assuming the null hypothesis is true, of obtaining a value of the test statistic at least as contradictory to H_0 as the value calculated from the available sample. (*Modern Mathematical Statistics with Applications*, p. 456)

The whole topic of the interpretation and use of p-values is complex and highly contested. In fact, it took 20 statisticians 2 days and many subsequent days of drafting to produce the American Statistical Association's statement on p-values: The statement by the American Statistical Association (Wasserstein and Lazar, 2016).

What p-values are We quote 2 of the 6 points in the statement here. Firstly, what p-values are:

> ***P*-values can indicate how incompatible the data are with a specified statistical model**...
> The smaller the p-value, the greater the statistical incompatibility of the data with the null hypothesis, if the underlying assumptions used to calculate the p-value hold. This incompatibility can be interpreted as casting doubt on or providing evidence against the null hypothesis or the underlying assumptions. (*ASA Statement on Statistical Significance and P-values*)

In the Swain versus Alabama example where we imagined there were 15 Black people on the jury, the small p-value ($p = 0.0062$) indicates that the data (here 15 Black people on the panel) are quite incompatible with the null hypothesis statistical model (here that Black and non-Black people were drawn from the population at random). The low p-value casts considerable doubt on the hypothesis. Of course the actual data ($t_0 = 8$) has a vanishingly small p-value (Table 17.1).

What p-values are not Secondly, what they are not:

> ***P*-values do not measure the probability that the studied hypothesis is true, or the probability that the data were produced by random chance alone.**
>
> Researchers often wish to turn a p-value into a statement about the truth of a null hypothesis, or about the probability that random chance produced the observed data. The p-value is neither. It is a statement about data in relation to a specified hypothetical explanation, and is not a statement about the explanation itself. (*ASA Statement on Statistical Significance and P-values*)

"Statistical significance" A widespread practice in scientific literature is to take p-values of less than $p = 0.05$ as indicating **statistical significance**, i.e. that the null hypothesis should be rejected. Values of less than 0.05 indicate weak evidence against the null hypothesis. Sometimes higher thresholds are used, e.g. $p = 0.01$ and $p = 0.001$. In scientific papers and the output from stats packages you will sometimes see these values indicated with asterisks:

- * means significant at least at the $p < 0.05$ level
- ** means significant at least at the $p < 0.01$ level
- *** means significant at least at the $p < 0.001$ level

There is no "correct" answer about what the right level of significance is. The $p < 0.05$ value was suggested in a paper by the statistician Ronald Fisher[1], who invented the hypothesis test, but it simply seemed "convenient" to him for his purposes. As in the discussion on confidence intervals (How big should a confidence interval be?), the value we choose to use may depend on the application. For example, we would demand a very low p-value when testing the null hypothesis that a new drug has no effect on the death rate of patients. We might accept a slightly higher p-value for the hypothesis that it has no positive effect on symptoms. In less mission-critical scientific applications, a higher p-value will be acceptable.

17.3 Testing for goodness of fit to a model

Multiple categories In the example so far, there have been just two categories: Black and non-Black. In 2010 the North California branch of the American Civil Liberties Union (ACLU) investigated the numbers of Caucasian, Black/African American, Hispanic, Asian/Pacific Islander and Other people on jury panels in Alameda County. The found the data shown in the first two rows of Table 17.2.

We want to test the following null and alternative hypotheses, which are essentially the same as for the case with two categories:

The null hypothesis H_0**:** The jury panels were chosen by random selection from the population in a district.

The alternative hypothesis H_a**:** The jury panels were chosen by some other, unspecified, method.

With two categories, it's easy to see that the number of Black people could be a test statistic. But in this case, there are 4 numbers that describe the outcome of any simulation (we can always compute the number in the 5th category if we know the total number and the numbers in 4 categories). We can't have 4 test statistics, so we need to create a statistic that indicates the disparity between the observed and expected outcomes.

[1]Fisher studied under Pearson, and developed a huge body of modern statistics. He also edited the *Annals of Eugenics* and had controversial views on race.

	Caucasian	Black/AA	Hispanic	Asian/PI	Other	Total
Population %	54	18	12	15	1	100
Observed panel numbers	780	117	114	384	58	1453
Expected panel numbers	784.62	261.54	174.36	217.95	14.53	1453.00
$\frac{(\text{Observed}-\text{Expected})^2}{\text{Expected}}$	0.03	79.88	20.90	126.51	130.05	357.36

Table 17.2: Alameda County jury panel data. The top row shows the estimated proportions of 5 ethnic groups (Caucasian, Black/African American, Hispanic, Asian/Pacific Islander and Other) in Alameda County. The second row (Observed panel numbers) shows the total number in each group on 11 jury panels from 2009–2010. There was a total of 1453 on the 11 jury panels (final column). The third row (Expected panel numbers) shows the numbers we would expect from each group if the panels had been selected randomly from the population. The final row $\frac{(\text{Observed}-\text{Expected})^2}{\text{Expected}}$ shows the disparity between the observed and expected using this formula. The total disparity is in the final column.

Suppose we call the population proportions of each of k groups p_i and the observed numbers in each group n_i. The total number sitting on jury panels is $n = \sum_i n_i$. We can compute the numbers we would expect to be on jury panels as np_i (third row of table). One measure of disparity would be the sum of the squared differences:

$$\sum_{i=1}^{k}(n_i - np_i)^2$$

This looks at the *absolute* squared differences between the expected and observed values for each category. If we expected $np_1 = 100$ in one category and observed $N_1 = 95$, this expected-observed pair would contribute 25 to the sum. A difference of $np_2 = 10$ (expected) and $N_2 = 5$ (observed) would also contribute 25 to the sum. However, in *relative* terms, the difference between the first expected-observed pair is 5%, whereas in the second pair it is 50%.

This motivates us to look at the scale the disparity measure by dividing by the expected number in each category, to create a statistic that we call **chi-squared**, written using the Greek symbol χ^2:

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - np_i)^2}{np_i} \tag{17.5}$$

The components of the χ^2 statistic are seen in the final row of Table 17.2, as is the value of $\chi^2 = 357.36$ for the observed values (in the “Total” column).

Statistical simulation We can now run a statistical simulation to generate the expected distribution of χ^2. For each repetition we simulate the numbers in each category by drawing from a multinomial distribution with parameters n and p_i. We then compute and store χ^2 for that simulation, which gives us the simulated distribution shown in blue in Figure 17.3. We can immediately see that the observed value of $\chi^2 = 357.36$ is off the scale of the graph, indicating that it has a much bigger value than is compatible with the null hypothesis, so we reject the null hypothesis.

Chi-squared distribution It turns out that, providing every expected value np_i is greater than 5, the χ^2 statistic is distributed approximately according to a χ^2 probability distribution with $k-1$ degrees of freedom, shown in orange in Figure 17.3. The fit between the probability distribution the simulated distribution is clear.

Goodness-of-fit Large values of χ^2 statistic (i.e. the upper tails of the distribution) indicate a poor **goodness-of-fit** between the model and the data. χ^2 tests therefore tend to be **upper-tailed**. However, the statistic had a very low χ^2, we might be suspicious that the data had been fiddled with.

[2]Sometimes you may see the letter X used instead of χ.

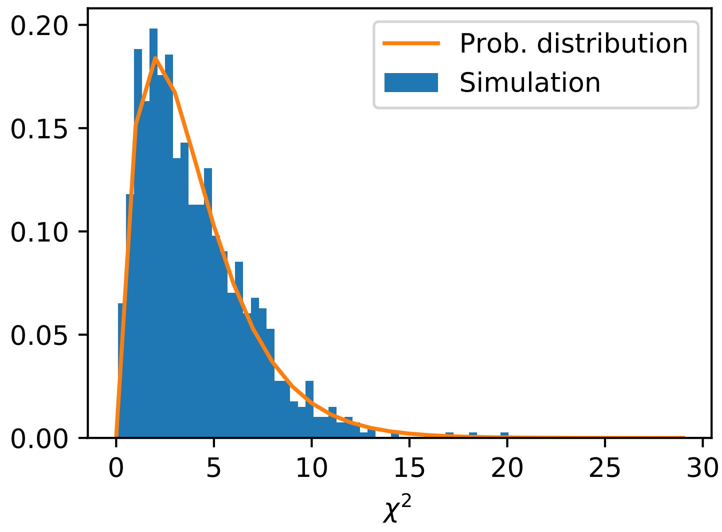


Figure 17.3: Distribution of χ^2 for jury panel selection in Alameda County. Simulations shown in blue and theoretical χ^2 distribution with 4 degrees of freedom shown in orange.

	Female	Male	Total
Depressed	30	12	42
Not depressed	2048	1663	3711
Total	2078	1675	3753

	Population 1	Population 2	Total
Category 1	n_{11}	n_{12}	$n_{1\bullet}$
Category 2	n_{21}	n_{22}	$n_{2\bullet}$
Total	$n_{\bullet 1}$	$n_{\bullet 2}$	$n_{\bullet\bullet}$

Table 17.3: Left: Contingency table of the number of depressed and not depressed people in a population of females and males; data based on a prospective study Bornioli et al. (2020). Right: General symbolic version of the two-way contingency table. There are I rows and J columns. The number of items falling into a cell in the ith row and jth column is denoted n_{ij}. The total in the ith row is denoted $n_{i\bullet} = \sum_{j=1}^{J} n_{ij}$, the total in the jth column is $n_{\bullet j} = \sum_{j=1}^{I} n_{ij}$. The grand total is $n_{\bullet\bullet} = \sum_i n_{i\bullet} = \sum_j n_{\bullet j}$.

The χ^2 statistic can be used to assess the goodness-of-fit of many types of model and data, not just this proportion example. If we find a χ^2 with a p-value greater than desired cut-off, this suggests that we should not reject the model.

Testing for independence with two-way contingency tables We may have multiple populations (e.g. males and females) and multiple categories (e.g. depressed or not depressed). We can arrange these in a **two-way contingency table** (Table 17.3).

We want to test the null hypothesis that being depressed is independent of if you are male of female. In other words $P(X = x, Y = y) = P(X = x)P(Y = y)$. Using a notation similar to Table 17.3 (right), we can write this probability as $p_{ij} = p_{i\bullet}p_{\bullet j}$, where $p_{i\bullet}$ is the marginal probability of an item being in category i and $p_{\bullet j}$ is the marginal probability of an item being in category j. Our best estimates of the marginal probabilities are

$$p_{i\bullet} = \frac{n_{i\bullet}}{n_{\bullet\bullet}} \text{ and } p_{\bullet j} = \frac{n_{\bullet j}}{n_{\bullet\bullet}} \tag{17.6}$$

Therefore the best estimates of the number of in each cell are

$$\hat{e}_{ij} = n_{\bullet\bullet}p_{ij} = \frac{n_{i\bullet}n_{\bullet j}}{n_{\bullet\bullet}} \tag{17.7}$$

	Female	Male
Depressed	23.25	18.75
Not depressed	2054.75	1656.25

	Population 1	Population 2
Category 1	$\hat{e}_{11}$	$\hat{e}_{12}$
Category 2	$\hat{e}_{21}$	$\hat{e}_{22}$

Table 17.4: Expected numbers in contingency table, in example (left) and in symbols from Equation 17.7 (right).

The χ^2 statistic is computed as

$$\chi^2 = \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}} = \sum_i \sum_j \frac{(n_{ij} - \hat{e}_{ij})^2}{\hat{e}_{ij}} \tag{17.8}$$

In this case it is 4.433.

We assume that the numbers of depressed and non-depressed, and males and females are fixed. In general, if there are I rows and J columns in the table there are $(I-1)(J-1)$ degrees of freedom. In this case there is therefore only 1 degree of freedom; specifying n_{11} (or any other cell) allows us to compute the values of all the other cells. We therefore look up the cumulative distribution function of χ^2 with 1 degree of freedom, to find that $p < 0.035$, so this is significantly different from independence at the 5% level.

17.4 Issues in hypothesis testing

Type I and Type II errors Regardless of whether we use a one-tailed test or a two-tailed test, *if* the null hypothesis were true, there is 5% chance that an observed test statistic in the rejection region might really have arisen by chance. By rejecting H_0, we would have made a **Type I error**: rejecting the null hypothesis when it is true. To reduce the risk of making a Type I error, we could make the rejection region smaller, e.g. the bottom 1%. However, we would also have increased the chance of making a **Type II error**: not rejecting the null hypothesis when it is false.

There is no right answer about what size of rejection region to use – it depends on what the consequences of Type I versus Type II errors are.

Decisions based on confidence intervals

> Scientific conclusions and business or policy decisions should not be based only on whether a *p*-value passes a specific threshold. (ASA Statement, point 5). (Wasserstein and Lazar, 2016)

$p \leq 0.05$ does not mean that the false; it is one point in a spectrum. However, it is often seen as the "holy grail" of scientific research.

Data snooping and *p*-value hacking It is very tempting to try out many experiments in order to get a *p*-value of less than 0.05. However, the more experiments are run, the more chance there is of Type I errors – i.e. rejecting the null hypothesis when it is true.

"Data snooping" or "*p*-value hacking" is the practice of rerunning experiments or selecting subsets of datasets until a statistically significant result is achieved. It is harder to publish negative results than positive results in academic journals, so there is an incentive to data snoop. Some statistically significant results in the literature will be Type I errors, which makes it important to replicate experimental results.

The ASA statement says:

> **Proper inference requires full reporting and transparency.**
>
> *P*-values and related analyses should not be reported selectively. Conducting multiple analyses of the data and reporting only those with certain *p*-values (typically those passing a significance threshold) renders the reported *p*-values essentially uninterpretable. Cherry-picking promising findings, also known by such terms as data dredging, significance chasing, significance questing, selective inference, and "p-hacking," leads to a spurious excess of statistically significant results in the published literature and should be vigorously avoided... (*ASA Statement on Statistical Significance and P-values*)

Multiple testing Suppose we undertake multiple tests on the same dataset is problematic, and find that one of the tests is significant. As we increase the number of tests, the probability of a Type I error increases (XCKD comic in reading). If we undertake 20 tests, there's a 0.95^{20} chance of not having a Type I error, and therefore a $1 - 0.95^{20} = 0.64$ chance of a type I error. There are ways to compute more stringent cut-offs in these cases, for example the Bonferroni correction.

Chapter 18

A/B testing

Recommended reading

- *Modern Mathematical Statistics with Applications*, Chapter 10

18.1 The principle of A/B Testing

A/B testing **A/B testing** is a method for assessing how changes to design of a system affect user behaviour. Figure 18.1 shows a hypothetical example, in which two versions of a website are presented to users selected at random. Group A gets the version with the blue button and group B gets the version with the green button with the inviting arrow. The numbers of users clicking-through is then measured. There are a number of commercial systems to implement A/B testing.

Statistical Questions in A/B testing

1. Is A *significantly* better than B?
2. How much better is A than B?

Generating confidence intervals for A/B learning using statistical simulations Let's imagine that we present the two versions of the page to group A and to group B the same number of times, n. We find that group A clicks through on 70% of occasions and group B on 72%. We'll call the underlying proportions of users that click through that we are trying to estimate p_A and p_B, and we will define the difference that we are trying to estimate:

$$d = p_\mathrm{A} - p_\mathrm{B} \tag{18.1}$$

The difference d is positive when A is better than B. We can address the question of how much better than A is than B by finding a point estimate of d – the larger d the better A is than B. We can address the question of if A is significantly better than B by finding a confidence interval.

The natural point estimators for p_A, p_B and d are:

$$\hat{p}_\mathrm{A} = \frac{n_\mathrm{A}}{n} \quad , \quad \hat{p}_\mathrm{B} = \frac{n_\mathrm{B}}{n} \quad \text{and} \quad \hat{d} = \hat{p}_\mathrm{A} - \hat{p}_\mathrm{B} \tag{18.2}$$

where n_A and n_B are the actual numbers clicking through from A and B.

To find the confidence interval, we can use a statistical distribution of d, assuming the underlying proportions in populations A and B are given by the point estimates $\hat{p}_\mathrm{A}$ and $\hat{p}_\mathrm{B}$. The routine to generate the sampling distribution of d looks like:

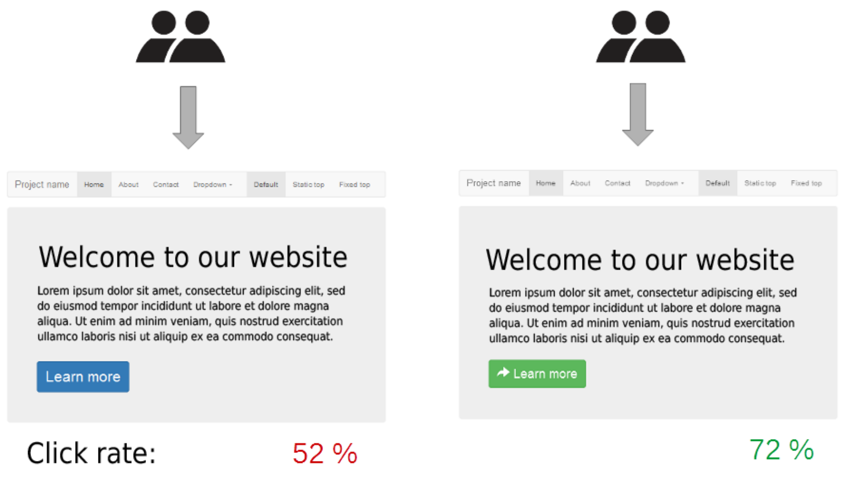

Project name
Home
About
Contact
Dropdown
Default
Static top
Fixed top
Welcome to our website
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Learn more
Click rate:
52 %
72 %

Figure 18.1: A/B Testing. Group A is shown the web page on the left; group B the one on the right. Image credit: Maxime Lorant, Wikimedia, CC SA 4.0.

- For j in $1, \dots, k$
  - Sample n^*_{A} from binomial distribution with parameters n and $\hat{p}_{\mathrm{A}}$
  - Sample n^*_{B} from binomial distribution with parameters n and $\hat{p}_{\mathrm{B}}$
  - Compute and store difference in proportions

$$d^*_j = n^*_{\mathrm{A}}/n - n^*_{\mathrm{B}}/n$$

- Plot the distribution of d^* and compute the desired quantities

The result is shown in Figure 18.2. The point estimate $\hat{d} = -0.02$, suggesting that B is better than A. However, the 95% confidence interval is $(-0.06, 0.02)$, which contains the value $d = 0$, suggesting that A and B could be equally effective.

Note that the area to the left of $\hat{d} = 0$ in the bootstrap distribution is about 85% and the area to the right is about 15%. We interpret this as meaning that there is an 85% chance that version B is better than version A – but there is still a 15% chance that it isn't.

Undertaking a hypothesis test for A/B learning using statistical simulations It's also possible to approach this A/B problem as a hypothesis test. We leave it as an exercise to formulate the problem in this way and write a statistical simulation.

18.2 Increasing certainty in A/B testing

Getting a more certain result To be more certain, we could keep the test running. But, assuming that the population proportions are $p_{\mathrm{A}} = 0.70$ and $p_{\mathrm{B}} = 0.72$ how many runs would we need to have a chance of (say) only 1% that A is better than B?

The brute force approach to find out how big n should be is to run the bootstrap again, with different values of n (Figure 18.3). As n increases, the distribution, and the 95% confidence interval gets narrower. By $n = 10000$, we can see that the upper end of the 99% confidence interval is now less than 0. The chance of the underlying proportion p_{A} being higher than p_{B} is around 0.0012. We can therefore say that a 99.999% confidence interval is $(-\infty, 0)$.

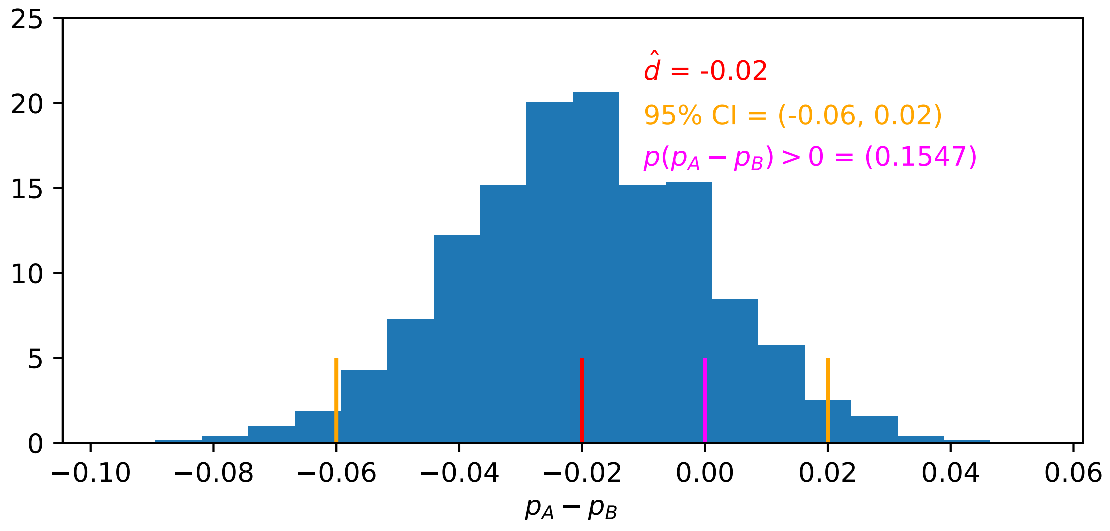


Figure 18.2: Bootstrap simulation of A/B test with $n = 1000$, $p_\text{A} = 0.70$ and $p_\text{B} = 0.72$.

When to stop sampling Suppose we had collected our first $n = 1000$ A and B responses in 2 hours on a Monday afternoon. We're quite excited by the result, and reckon that we need to keep it running up to $n = 10000$ in order to be 99.999% certain. This will probably take us to Tuesday afternoon, we'll then write a report for the boss, and be done by Wednesday. What could possibly go wrong?

We've made a hidden assumption that every period of the week is like a Monday afternoon. What if people prefer blue to green in the evening? What if the Monday afternoon demographic is older, but the weekend demographic is younger? We may wish to collect at least a full week of data to check that our result really is robust – a week's worth of data should mean that any day- or time-specific effects are eliminated, or at least greatly reduced.

18.3 Large sample theory of A/B testing

As with the confidence intervals and hypothesis testing for the sample mean, we can use a theoretical approach to determine confidence intervals or undertake hypothesis testing when doing A/B testing. We have already determined that the estimators for the population proportions are:

$$\hat{p}_\text{A} = \frac{n_\text{A}}{n} \quad ; \quad \hat{p}_\text{B} = \frac{n_\text{B}}{n} \tag{18.3}$$

Now we are interested in estimating the difference $d = p_\text{A} - p_\text{B}$ between our population proportions. An unbiased estimator of d is:

$$\hat{d} = \hat{p}_\text{A} - \hat{p}_\text{B} \tag{18.4}$$

Supposing the population proportions are p_A and p_B, we expect the number of successes in n trials to be binomially distributed, with the standard deviations of n_A and n_B being:

$$\sigma_{n_\text{A}} = \sqrt{np_\text{A}(1 - p_\text{A})} \quad ; \quad \sigma_{n_\text{B}} = \sqrt{np_\text{B}(1 - p_\text{B})} \tag{18.5}$$

Dividing through by n and replacing p_A and p_B by their estimates, we get the estimated standard errors of the estimators $\hat{p}_\text{A}$ and $\hat{p}_\text{B}$:

$$\hat{\sigma}_{\hat{p}_\text{A}} = \sqrt{\frac{\hat{p}_\text{A}(1 - \hat{p}_\text{A})}{n}} \quad ; \quad \hat{\sigma}_{\hat{p}_\text{B}} = \sqrt{\frac{\hat{p}_\text{B}(1 - \hat{p}_\text{B})}{n}} \tag{18.6}$$

Since the samples from A and B are independent, the variance of the estimator of the difference in proportions $\hat{d}$ is equal to the sum of the variances of $\hat{p}_\text{A}$ and $\hat{p}_\text{B}$. We take the square root to get the standard error of the

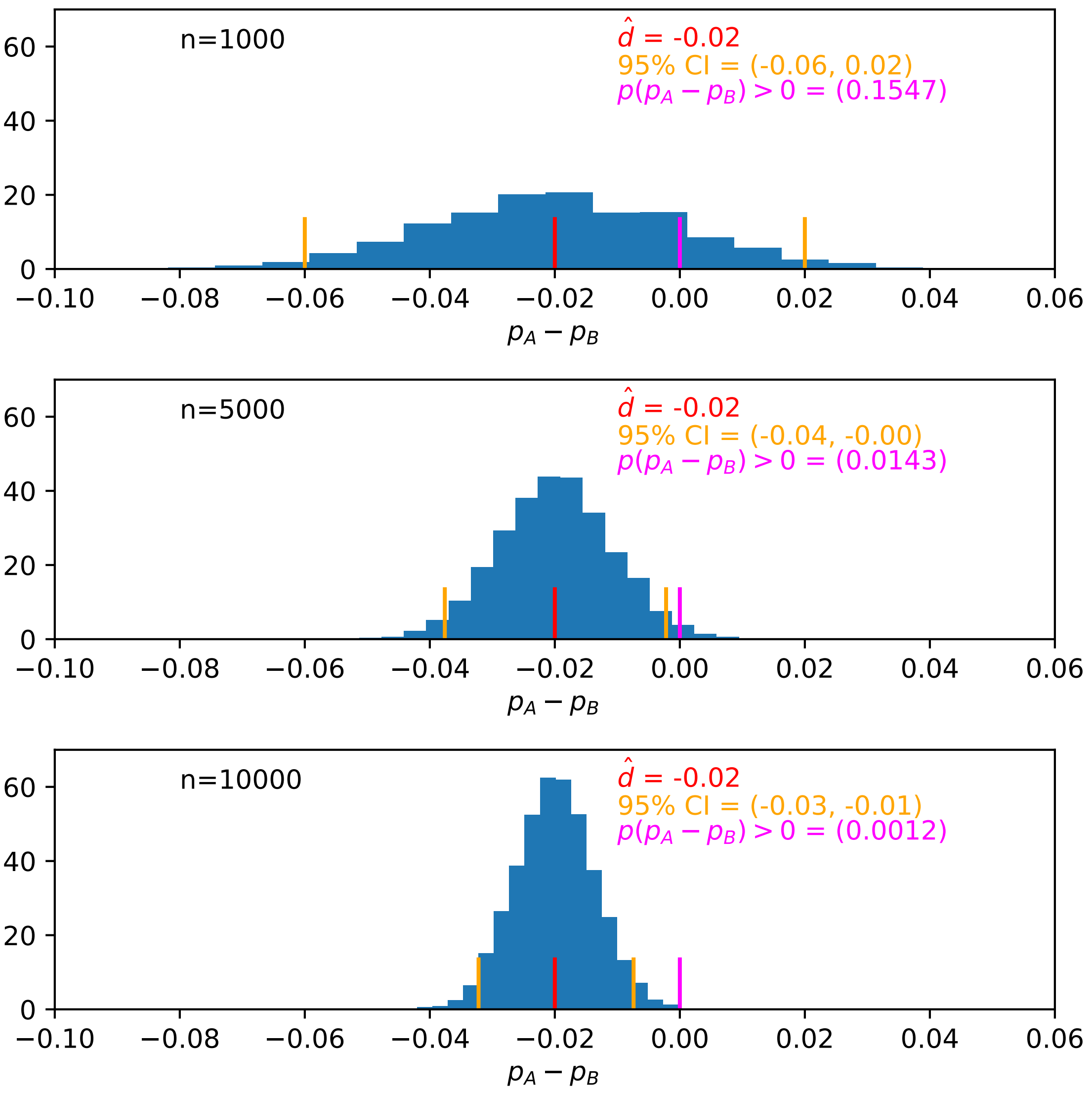


Figure 18.3: Bootstrap simulation of A/B test with $p_A = 0.70$ and $p_B = 0.72$, and varying numbers of n.

estimator $\hat{d}$:

$$\hat{\sigma}_{\hat{d}} = \sqrt{\hat{\sigma}^2_{\hat{p}_{\mathrm{A}}} + \hat{\sigma}^2_{\hat{p}_{\mathrm{B}}}} = \frac{\sqrt{\hat{p}_{\mathrm{A}}(1-\hat{p}_{\mathrm{A}}) + \hat{p}_{\mathrm{B}}(1-\hat{p}_{\mathrm{B}})}}{\sqrt{n}} \tag{18.7}$$

We'll assume that n is large, in which case the Central Limit Theorem applies, and we can assume that there is little variance in the estimated standard error of $\hat{d}$. We can therefore assume that the statistic

$$Z = \frac{\hat{d} - d}{\hat{\sigma}_{\hat{d}}} = \frac{(\hat{p}_{\mathrm{A}} - \hat{p}_{\mathrm{B}}) - (p_{\mathrm{A}} - p_{\mathrm{B}})}{\sqrt{(\hat{p}_{\mathrm{A}}(1-\hat{p}_{\mathrm{A}}) + \hat{p}_{\mathrm{B}}(1-\hat{p}_{\mathrm{B}}))/n}} \tag{18.8}$$

is normally distributed. We can then use the z-distribution to calculate confidence intervals.

Worked example We'll use figures we used for the bootstrap to find the 95% confidence interval theoretically:

$$\hat{d} = \hat{p}_{\mathrm{A}} - \hat{p}_{\mathrm{B}} = 0.70 - 0.72 = -0.02 \tag{18.9}$$

$$\hat{\sigma}_{\hat{d}} = \frac{\sqrt{\hat{p}_{\mathrm{A}}(1-\hat{p}_{\mathrm{A}}) + \hat{p}_{\mathrm{B}}(1-\hat{p}_{\mathrm{B}})}}{\sqrt{n}} = \frac{\sqrt{0.70(1-0.70) + 0.72(1-0.72)}}{\sqrt{1000}} = 0.020 \tag{18.10}$$

For a 95% confidence interval (which makes sense here), we need to use the z critical value $z_{0.025} = 1.96$. The confidence interval for $\hat{d}$ is therefore

$$\begin{aligned}&(\hat{d} - z_{0.025}\hat{\sigma}_{\hat{d}}, \hat{d} + z_{0.025}\hat{\sigma}_{\hat{d}})\\ =&(-0.02 - 1.96 \times 0.020, -0.02 + 1.96 \times 0.020)\\ =&(-0.60, 0.20)\end{aligned} \tag{18.11}$$

This is almost exactly the same as the bootstrap estimates.

18.4 Issues in A/B testing

Statistical versus practical significance in A/B testing There is an important distinction between statistical significance and practical significance. We might test the run time of two versions (A and B) of a webserver program on random hits from users. In the example that we've just seen if we make n large enough, we can show that there B is better than A with a p-value of 0.001. However, is the difference of 2% actually that meaningful? In this case it is still probably worth it, since it requires little or no extra effort or energy to create a green button rather than a blue button. But if the processing required to serve version B used a lot more energy, maybe that 2% improvement wouldn't be worth it.

Quoting from the ASA statement on p-values again:

> A p-value, or statistical significance, does not measure the size of an effect or the importance of a result. Statistical significance is not equivalent to scientific, human, or economic significance. Smaller p-values do not necessarily imply the presence of larger or more important effects, and larger p-values do not imply a lack of importance or even lack of effect. Any effect, no matter how tiny, can produce a small p-value if the sample size or measurement precision is high enough, and large effects may produce unimpressive p-values if the sample size is small or measurements are imprecise. Similarly, identical estimated effects will have different p-values if the precision of the estimates differs. (Wasserstein and Lazar, 2016)

Ethical questions in A/B testing There are a number of ethical issues we should consider in A/B testing:

- We are undertaking experiments on people
- In a commercial situation, users do not give informed consent.
- In an academic situation, informed consent is required – but how can we get this informed consent without affecting the experiment?

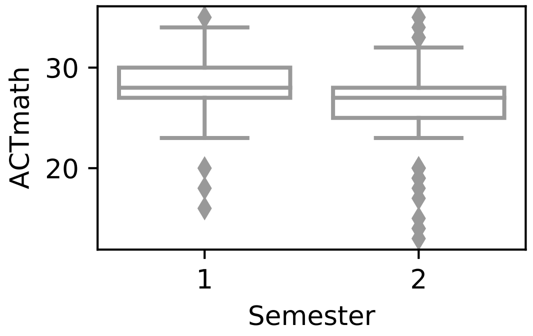


Figure 18.4: Maths scores in Semester 1 (Autumn) and Semester 2 (Spring). Data from Edge and Friedberg (1984), via Devore and Berk (2012).

- What about data protection?

The experiment in which Facebook manipulated news feeds to explore the effect on users' moods (Kramer et al., 2014) was an example of A/B testing that was widely seen as problematic (Verma, 2014) because of a lack of informed consent, no opportunity to opt-out and no institutional review of the experiment, because it was carried out by a private company. Many A/B tests are arguably not having such a significant effect on users – but they may have some effect nonetheless.

It's therefore important to reflect on an A/B test before setting it up. Questions might include:

- Would I feel comfortable if this change was tested on me?
- What potential harms could be caused to users?

Comparing two samples more generally The methods we show here predate A/B testing, and come under the heading of "comparing two samples". Sometimes our groups may have numeric response variables, and need not be from a controlled situation – for example the scores of students on a calculus course in Semester 1 (Group A) and Semester 2 (Group B, Figure 18.4). There are a number of variations on the theoretical method presented here for generating confidence intervals and undertaking hypothesis tests. Which one to use depends on whether the data is approximately normal, how large the sample size is and if the data comes from a set of paired measurements, e.g. pairs of measurements of temperature at dawn and dusk from a number of different locations. *Modern Mathematical Statistics with Applications* chapter 10 has many more details, and we go through the maths example below.

18.5 Samples with numeric responses

The problem of two samples In A/B testing the samples A and B comprise n binary response variables, and we estimated the difference in proportions between the groups. A related problem is when the response variables are numeric rather than binary, and we wish to assess whether the difference between distributions of a numeric variable recorded in two populations is similar or different. The different populations may arise from a randomised controlled trial, in which participants are assigned randomly to a control group or a treatment group. However, we can also test differences between two observed groups.

For example, the groups A and B could be students taking a calculus course in Semester 1 (Group A) and students taking the course in Semester 2 (Group B). Figure 18.4 plots these distributions using boxplots. The maths scores of students taking a calculus course in Semester 2 seem to be lower than the grades in Semester 1.

We'll call the grades of the m Semester 1 students $x_1, \dots, x_m$ and the grades in the n Semester 2 students $y_1, \dots, y_n$. We can compute the two means of the grades of each group, and find that difference is $\overline{x} - \overline{y} = 2.37$. But, assuming that the maths scores are representative of performance in Semester 1 and Semester 2 in other

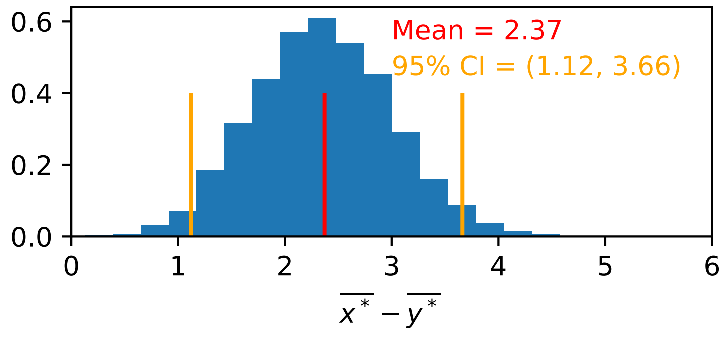


Figure 18.5: Bootstrap distribution of the maths grades example.

years, we'd like to find a 95% confidence interval for the difference – in other words an interval that we would expect to contain the true, underlying difference 95% of the time.

We already know how to compute a confidence interval of one sample, using the bootstrap. Can we adapt it to give us a confidence interval around the difference between two means?

Applying the bootstrap To apply the bootstrap, on each bootstrap step we sample with replacement from both groups:

- For j in $1, \dots, B$
  - Take sample x^* of size m from the sample *with replacement*
  - Take sample y^* of size n from the sample *with replacement*
  - Compute the sample mean of the new samples, $\overline{x_j^*}$ and $\overline{y_j^*}$
  - Compute and store the difference in the sample means $\overline{x_j^*} - \overline{y_j^*}$
- Plot the bootstrap distribution of $\overline{x_j^*} - \overline{y_j^*}$
- We can also compute the bootstrap estimator of the variance of the difference between the mean:

$$s^2_{\text{boot}} = \frac{\sum_{j=1}^{B} (\overline{x_j^*} - \overline{y_j^*})^2}{B - 1}$$

The bootstrap distribution is shown in Figure 18.5. We can see that the 95% confidence interval is 1.11 to 3.66.

18.6 Relationship between hypothesis testing and confidence intervals

Suppose we had wanted to test the hypothesis that average performance in Semester 2 is different to average performance in Semester 1. Our null hypothesis would be

H_0: The mean performance in semester 1 is the same as the mean performance in semester 2.

The alternative hypothesis would be:

H_a: The mean performance in semester 1 is different from the mean performance in semester 2.

We've previously simulated the null hypothesis (here, no difference in means) to generate a distribution of what the test statistic (here, the difference in the sample means) would be under the null hypothesis, and then compared this distribution with the observed value of our test statistic.

It turns out that there is a duality between confidence intervals and hypothesis testing. Instead of the distribution of the sampling distribution generated under the null hypothesis, we have used the observed data to generate the distribution of the estimator for the parameter corresponding to the test statistic. Instead of the observed value of the test statistic, we have the value the parameter would take under the null hypothesis.

In this example, we used the bootstrap to estimate the distribution of the estimator of the difference of the means $\mu_x - \mu_y$ (Figure 18.5), and we could then ask if the null hypothesis value of the difference in the means (0) lies in either of the rejection regions in the tails of that distribution. If so, we can reject at the level corresponding to the size of the tails. Alternatively, we could compute a p-value, by finding at what quantile the null hypothesis value (here 0) lies on the distribution. In this case we would find $p = 0$, so the null hypothesis would be rejected.

For more on the duality between confidence intervals and hypothesis testing see this Quora article.

18.7 Paired data

18.8 The theoretical method of testing for differences between groups

Samples with known variance We can also undertake hypothesis testing and compute confidence intervals for the difference between the means of two samples theoretically. The assumptions are:

- $X_1, \dots, X_m$ is a random sample from a population with mean μ_1 and variance σ_1^2
- $Y_1, \dots, Y_n$ is a random sample from a population with mean μ_2 and variance σ_2^2
- The samples are independent of each other.

Estimator of the difference The difference between the sample means $\overline{X} - \overline{Y}$ is an unbiased estimator of the difference between the true means $\mu_1 - \mu_2$. This follows from $\overline{X}$ being an unbiased estimator of μ_1 and $\overline{Y}$ being an unbiased estimator of μ_2. The standard deviation of the estimator is

$$\sigma_{\overline{X}-\overline{Y}} = \sqrt{\frac{\sigma_1^2}{m} + \frac{\sigma_2^2}{n}} \tag{18.12}$$

This follows from the two samples being independent, so $V(\overline{X}) - V(\overline{Y}) = V(\overline{X}) + V(\overline{Y}) = \sigma_1^2/m + \sigma_2^2/n$.

Theoretical Distribution for large samples As we with the sample mean of one population, we define a standardised test statistic, which we expect to be zero in the case of a null hypothesis that the true difference between the population means is $\mu_1 - \mu_2$:

$$Z = \frac{\overline{X} - \overline{Y} - (\mu_1 - \mu_2)}{\sqrt{S_1^2/m + S_2^2/n}} \tag{18.13}$$

The denominator is the sample standard deviation of the estimator, and is a random variable. As with the sample of one mean, for small m and n this will vary considerably between different samples, and so we do have to consider these random effects. However, for large m and n, it will approximate the true population means, and its variability is low enough to consider it as a fixed parameter. In the limit of large n and m, the central limit theorem suggests that the distribution of the statistic should be normal, so we can use a z-test.

Theoretical Distribution for small samples In the case of smaller samples, the variability of the sample standard deviation has to be taken into account, and it turns out that the sampling distribution of the standardised statistic is a t-distribution with a number of degrees of freedom ν that depends on the standard deviations of both distributions:

$$\nu = \frac{\left(s_1^2/m + s_2^2/n\right)^2}{\frac{(s_1^2/m)^2}{m-1} + \frac{(s_2^2/n)^2}{n-1}} \tag{18.14}$$

To determine a confidence interval of $1-\alpha$, we can find the t critical value $t_{\alpha/2,\nu}$ which we will expect will contain the mean difference on $1-\alpha$ percent of replications. We can then use the t critical value to transform back to the unstandardised variables using the standard error of the difference of the mean:

$$\overline{x} = \overline{y} \pm t_{\alpha/2,\nu}\sqrt{s_1^2/m + s_2^2/n} \tag{18.15}$$

Related Workshop: Statistical problems 2

18.9 Non-examinable: Bayesian inference applied to A/B testing

So far we've approached statistics from a frequentist or sampling theory perspective. This is predicated on there being a real population parameter that we're trying to estimate, and then using the sampling distribution to model what happens in the process of taking a sample from the population. When we test hypotheses, we find the fraction of sampling simulations of the null hypothesis that could produce a result at least as extreme as what we observe, the test statistic. The two hypothesis that are competing are the specified null hypothesis and the alternative hypothesis, which is the "not the null hypothesis".

In Bayesian stats, we do not assume that a single population parameter exists. In the situation we have just described, this seems appropriate: there is a potentially infinite population of users, and so p_A and p_B do not exist in the same way that the number of wildcats in Scotland does.

We will can also use Bayesian stats to compare competing hypotheses. In this case our hypotheses might be that A has a higher click through rate than B, or vice versa.

A fundamental quantity in Bayesian statistics is the **posterior probability**, that is the probability distribution of a parameter (e.g. p_A or p_B) *after* we have observed data (e.g. $\hat{p}_\text{A}$ and $\hat{p}_\text{B}$ from n observations). The posterior probability is derived from the **likelihood** of generating an observation $\hat{p}_\text{A}$ given a value of the parameter p_A and the **prior probability** distribution.

The prior distribution allows us to express our beliefs about the likely distribution parameter should be *before* we have seen the data. This sounds like we are biasing the outcome – which we are – but this can make sense, e.g. we probably believe that a coin has a probability of 1/2 of landing on Heads – we would like a lot of data to convince us otherwise. However, if we really don't have much idea, we can set the prior distribution to be uniform.

Mathematically, Bayes theorem relates the posterior probability (called "the posterior" for short), the likelihood and the prior probability (called "the prior" for short). For our particular example for the A group, we'll consider the number of observed click-throughs $n_\text{A} = n\hat{p}_\text{A}$. In this case Bayes' theorem reads:

$$p(p_\text{A}|n_\text{A}, n) = \frac{p(n_\text{A}|p_\text{A}, n)p(p_\text{A})}{\int_{p_\text{A}=0}^{1} p(n_\text{A}|p_\text{A}, n)p(p_\text{A})\mathrm{d}p_\text{A}} \tag{18.16}$$

The denominator is the "probability of the data" $p(n_\text{A})$ and derives from the definition of conditional probability.

Suppose our model of how the data arises is that users decide, independently of each other, to click through with a probability p_A To start of with, we don't know what p_A is – we assume it is equally likely to be any number between 0 and 1. In other words, we are assuming a uniform prior: $p(p_\text{A}) = 1$ for $0 \le p_\text{A} \le 1$.

Probability theory tells us that if we have n repeats of a trial in which the probability of "success" on each trial is p_A, then the distribution of the total number of successes is given by a binomial distribution.

$$p(n_\text{A}|p_\text{A}, n) = \begin{pmatrix} n \\ n_\text{A} \end{pmatrix} p_\text{A}^{n_\text{A}}(1-p_\text{A})^{n-n_\text{A}} \tag{18.17}$$

This is the likelihood of observing n_A out of n users clicking through, given a hypothetical click-through probability of p_A.

Using integration by parts and recursion, we can evaluate the integral in the denominator to get $1/(n+1)$. We thus find that the posterior distribution is

$$p(p_\text{A}|n_\text{A}, n) = (n+1)\begin{pmatrix} n \\ n_\text{A} \end{pmatrix} p_\text{A}^{n_\text{A}}(1-p_\text{A})^{n-n_\text{A}} \tag{18.18}$$

Note that, although this looks like a binomial distribution, the variable is actually p_{A}, and this is therefore a beta distribution with $a = n_{\mathrm{A}} + 1$ and $b = n - n_{\mathrm{A}} + 1$[1]

This distribution is now the distribution of the parameter p_{A} given the data. Using differentiation, we can prove that the distribution has a maximum at $\hat{p}_{\mathrm{A}} = n_{\mathrm{A}}/n$. This makes sense, since we would expect the most likely value to be the observed proportion. The 2.5% and 97.5% centiles will enclose 95% of the distribution and are our Bayesian credibility interval – the analogue of a frequentist confidence interval.

We can write a similar expression for p_{B}. Since the A and B group are independent, the likelihood is

$$p(n_{\mathrm{A}}, n_{\mathrm{B}}|p_{\mathrm{A}}, p_{\mathrm{B}}, n) = \left(\begin{array}{c} n \\ n_{\mathrm{A}} \end{array} \right) p_{\mathrm{A}}^{n_{\mathrm{A}}}(1 - p_{\mathrm{A}})^{n-n_{\mathrm{A}}} \left(\begin{array}{c} n \\ n_{\mathrm{A}} \end{array} \right) p_{\mathrm{A}}^{n_{\mathrm{A}}}(1 - p_{\mathrm{A}})^{n-n_{\mathrm{A}}} \tag{18.19}$$

It turns out that the posterior $p(p_{\mathrm{A}}, p_{\mathrm{B}}|n_{\mathrm{A}}, n_{\mathrm{B}}, n)$ is also the product of the two posterior distributions.

$$p(p_{\mathrm{A}}, p_{\mathrm{B}}|n_{\mathrm{A}}, n_{\mathrm{B}}, n) = (n+1) \left(\begin{array}{c} n \\ n_{\mathrm{A}} \end{array} \right) p_{\mathrm{A}}^{n_{\mathrm{A}}}(1 - p_{\mathrm{A}})^{n-n_{\mathrm{A}}} \left(\begin{array}{c} n \\ n_{\mathrm{B}} \end{array} \right) p_{\mathrm{B}}^{n_{\mathrm{B}}}(1 - p_{\mathrm{B}})^{n-n_{\mathrm{B}}} \tag{18.20}$$

To compute the posterior distribution of the difference, we substitute $p_{\mathrm{B}} = p_{\mathrm{A}} - d$ and then integrate over p_{A} from d to 1 (if d is positive) or from 0 to $1 + d$ (if d is negative).

[1]We can verify the constant is correct:

$$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} = \frac{\Gamma(n+2)}{\Gamma(n_{\mathrm{A}}+1)\Gamma(n-n_{\mathrm{A}}+1)} = \frac{(n+1)!}{n_{\mathrm{A}}!(n-n_{\mathrm{A}})!}$$

Part V

Regression and inference

Chapter 19

Logistic regression

19.1 Principle of logistic regression

What is logistic regression used for? Logistic regression has various uses, including:

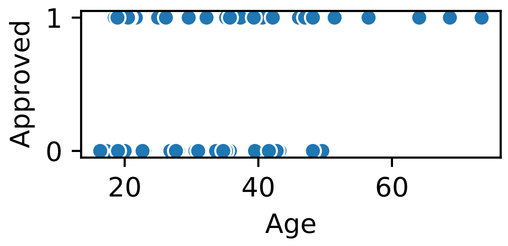
- **As a parametric supervised classification algorithm.** For example, a bank has data on previous customers it has considered offering credit cards to, including predictor variables (independent variables) such as their age, income, housing status and employment status. Each of these sets of variables is labelled with the response variable of whether the credit card was approved. The task is to determine if the credit card should be approved for a new customer.

- **As a way of investigating the association between predictor variables and a binary (also called dichotomous) response variable.** For example, suppose we have an observational study of patients of different ages, health levels, ethnicity and gender. Some of the patients have had a dose of vaccine for an illness, and some haven't. We'd like to know how the probability of getting the illness depends on if the vaccine has been administered or not. Like multiple linear regression, we can examine logistic regression coefficients to isolate the effect of the vaccine, controlling for the other variables.

Similarities and differences to k-NN We've already discussed classifiers, when we looked at k-Nearest Neighbours (Supervised learning: Classification with Nearest neighbours). As a reminder, the problem of classification is to predict the correct label for an unlabelled input item described by a feature vector of variables. As well as acting as a classifier, logistic regression can predict a real-valued number, the *probability* of a data point belonging to a category, on the basis of the predictors/predictor variables. In fact, we convert the logistic regression model into a classifier by choosing at a threshold level of probability at which we make a decision. For example, we might only want to approve cards that we think would have a 60% chance of being approved historically.

Association between continuous predictor and binary outcome We will use the example of the credit card approval to illustrate how logistic regression is used as a classifier and as a way of exploring associations between variables. Figure 19.1 visualises the relationships between age and approval and between employment status and approval in two ways. Because age is a continuous variable, we can plot individual datapoints on a scatter plot (Figure 19.1a). It looks like older customers were more likely to have their credit approved than younger ones.

Association between binary predictor and binary outcome: Odds and odds ratios Employment is a binary variable ("employed" or "not employed"). If we tried plotting it in the same way as age versus approval, we'd end up with a very uninformative plot, so instead we look at a **contingency table** (Figure 19.1b), which shows the empirical probability (relative frequency) of having credit approved or not approved based on employment status.



(a) Age versus approval. Each datapoint represents the age of a customer and whether their credit was approved (1) or not approved (0). It looks like a greater fraction of older customers had their credit approved than younger ones. A random subsample of data points is plotted to aid visualisation.

Employed	Approved	Not approved	Approval odds
0	0.25	0.75	0.34
1	0.71	0.29	2.42

(b) Contingency table showing empirical probabilities of approval ("Success") or not approval ("Failure") based on employment status (first two columns) and the odds of approval (final column). The odds are the probability of approval divided by the probability of not being approved.

Figure 19.1: Relationship between age, employment status and credit approval in the credit approval dataset.

In logistic regression, we will see that it makes sense to describe these probabilities in terms of **odds**[1], which we define as:

$$\text{Odds(Success)} = \frac{P(\text{Success})}{P(\text{Failure})} = \frac{P(\text{Success})}{1 - P(\text{Success})} \tag{19.1}$$

If success and failure are equally likely, the odds are equal to 1.

We call the **odds ratio** (OR) the ratio between the odds of credit approval if employed versus credit approval if not employed.

$$\text{OR}(x) = \frac{\text{Odds}(\text{Success}|x = \text{True})}{\text{Odds}(\text{Success}|x = \text{False})} \tag{19.2}$$

We can find the odds ratio of employment in the credit example by setting "Success" to "Approved" and x to "Employed", giving an answer of $7.09 = 2.42/0.34$. Thus, the odds of someone who is employed having credit approved are 7.09 times larger than the odds of someone who is not employed having credit approved. The odds ratio is sometimes referred to as an **effect size** and expressed as the percentage change in the odds from x being False to True; this case the effect size is 609%, since the effect of employment increases the odds of approval by this amount.

Principle of logistic regression with one predictor variable As its name suggests, logistic regression is related to linear regression. Suppose that the response variable (or dependent variable) y is a dichotomous variable (i.e. a categorical variable with two categories). We'll represent the categories by 0 (failure) and 1 (success). We'd like to model the probability $P(Y = 1|X = x)$ that the response variable is 1, given the predictor variable (or predictor) X has a value x. Because we're predicting a probability, the answer given by logistic regression has to lie between 0 and 1. Therefore, $P(Y = 1|X = x)$ can't be a linear function of x.

We get around this problem by allowing $P(Y = 1|X = x)$ to be a *nonlinear* function of x. A function that works well in many applications is the **logistic function**[2] (Figure 19.2). Using f to denote the logistic function,

[1] *Modern Mathematical Statistics with Applications* calls the "odds" the "odds ratio", which is not standard usage.

[2] Not-examinable: The logistic function is also known as the sigmoid function, and denoted $S(x)$ or $\sigma(x)$, due to its S-shaped curve. However, the term "sigmoid function" can refer to a family of S-shaped functions.

The term *logisitique* was first used in 1845 by Verhulst (1845) to describe the solution of a differential equation describing population

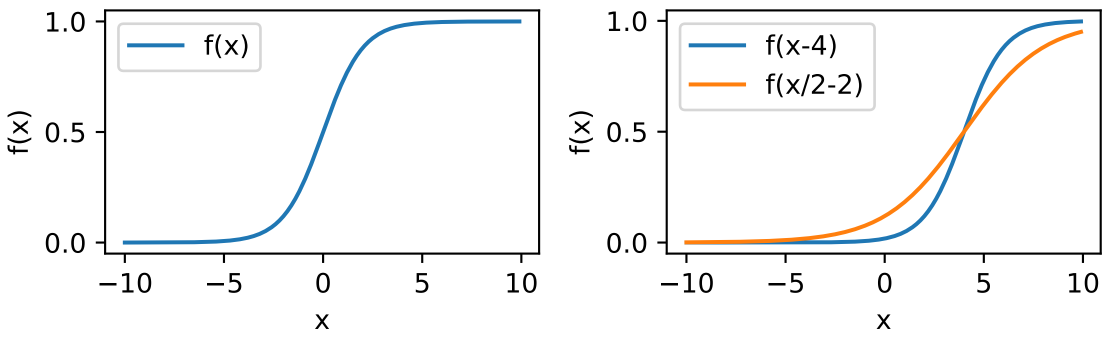


Figure 19.2: The logistic function. Left: the standard logistic function: $f(u) = \exp(u)/(1 + \exp(u))$, which can also be written $f(u) = 1/(1 + \exp(-u))$. Right: Examples of shifted logistic curves.

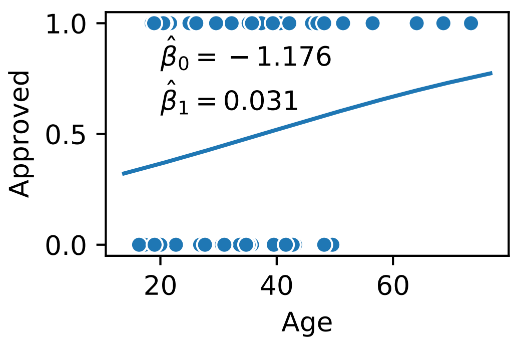


Figure 19.3: Logistic regression of age on credit approval.

the probability of a success is:

$$P(Y = 1|X = x) = f(\beta_0 + \beta_1 x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} = \frac{1}{1 + e^{-\beta_0-\beta_1 x}} \tag{19.3}$$

Just as with linear regression, we can adjust the values of the coefficients β_0 and β_1 to fit the data as best as possible – we will come to how we do this later.

Application to credit example with one variable Figure 19.3 shows the logistic regression of age on credit approval – we are ignoring all the other variables for now. The curve doesn't look very much like a logistic curve, but that's because it's got a very shallow slope, since $\hat{\beta}_1 = 0.03$. We can see that the probability ranges between about 0.37 for teenagers and 0.8 for 70-year-olds.

19.2 Interpretation of logistic regression coefficients

Interpretation of $\hat{\beta}_0$ In linear regression $\hat{\beta}_0$ is the intercept: it tells us the predicted value of the response variable when the predictor variable is 0. In logistic regression $f(\hat{\beta}_0) = 1/(1 + \exp(\hat{\beta}_0))$ tells us the probability

growth: $\frac{dp}{dt} = p(1 - p)$. However, Verhulst applied the term *logisitique* to the expression of time in terms of population, i.e. essentially $t = \ln(p/(1 - p))$. This is in fact the "logit" function: $\text{logit}(p) = \ln(p/(1 - p))$, which is the *inverse* of the logistic function. The name logit was coined much later – see later footnote.

In python `scipy` and some R packages, the logistic function is referred to as "expit", making "expit" the inverse of "logit", just as "log" is the inverse of "exp".

the response variable being 1 ("success") when the predictor variable is 0. In the credit example it suggests the likelihood of a newborn baby receiving credit approval is $f(-1.176) = 0.236$ – which seems rather high!

Log odds Remember the definition of odds (Equation 19.1). To interpret the coefficient $\hat{\beta}_1$ it helps to rewrite the logistic regression model (Equation 19.3) in terms of **log odds**, i.e. the log of the odds:

$$\text{Log Odds(Success)} = \ln \frac{P(\text{Success})}{P(\text{Failure})} = \ln \frac{P(\text{Success})}{1 - P(\text{Success})} \tag{19.4}$$

Log odds of 0 mean that success and failure are equally likely: $P(\text{Success}) = P(\text{Failure}) = 0.5$. Positive log odds mean that success is more likely than failure, and vice versa for negative log odds. An increase of 1 unit of the log odds means that the odds increase by a factor of e. As the probability tends towards 1, the log odds tend towards infinity; as the probability tends towards 0, the log odds tend towards negative infinity.

When we express probability in terms of log odds, we sometimes say it has units of "logits", which stands for *log*istic un*it*s. Going back to the example, we can say that when the predictor variable is 0, the log odds of approval are $\hat{\beta}_0 = -1.176$ logits.

The logit function converts the probability of success into the log odds of success to failure[3]:

$$\text{logit}(p) = \ln \frac{p}{1-p} \tag{19.5}$$

We can now re-express Equation 19.4 as Log Odds(Success) = logit(P(Success)).

Rewriting the logistic regression model in terms of log odds The probability of a failure is:

$$P(Y = 0|X = x) = 1 - f(\beta_0 + \beta_1 x) = 1 - \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} = \frac{1}{1 + e^{\beta_0+\beta_1 x}} = f(-\beta_0 - \beta_1 x) \tag{19.6}$$

We can divide Equation 19.3 by Equation 19.6 to obtain[4]:

$$\frac{P(Y = 1|X = x)}{P(Y = 0|X = x)} = \frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)} = e^{\beta_0+\beta_1 x} \tag{19.7}$$

The ratio on the left is the odds for success. It tells us how many times more likely the "success" ($Y = 1$) is than the "failure" ($Y = 0$) for any value of x (see Equation 19.1). If we take natural logs of both sides of the equation, we see that the log odds is a linear function of the predictor:

$$\text{logit}(P(Y = 1|X = x)) = \ln \frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)} = \ln \frac{P(Y = 1|X = x)}{P(Y = 0|X = x)} = \beta_0 + \beta_1 x$$

We can now see that $\hat{\beta}_0$ is the log odds when the predictor variable is equal to 0.

Interpretation of $\hat{\beta}_1$ From Equation 19.2, we can see that the parameter β_1 tells us the increase in the log odds when we increase x by 1 unit.

In other words, when we increase x by 1 the odds multiply by a factor $\exp(\beta_1)$. We refer to this factor as the odds ratio (OR) for the variable x. In this example the OR = exp(0.03) = 1.03. Thus, for every year of age, you're 1.03 times more likely to have a loan approved, an effect size of 3%.

[3]If we have a continuous response variable between 0 and 1 (e.g. the proportion p of organisms killed by a toxin), we could transform the response variable into logits using logit(p). In fact, logistic regression and the term logit were invented to deal with this sort of data (Berkson, 1944).

[4]This identity should help to see this:

$$\frac{f(u)}{f(-u)} = \frac{e^u}{1 + e^u} \frac{1 + e^{-u}}{e^{-u}} = e^u$$

	Variable	Coefficient	Odds or OR
$\hat{\beta}_0$	Intercept	-1.969	0.140
$\hat{\beta}_1$	Age	0.029	1.030
$\hat{\beta}_2$	Employed	1.881	6.562

Table 19.1: Coefficients expressed in raw form and as odds ratio $\exp(\beta)$.

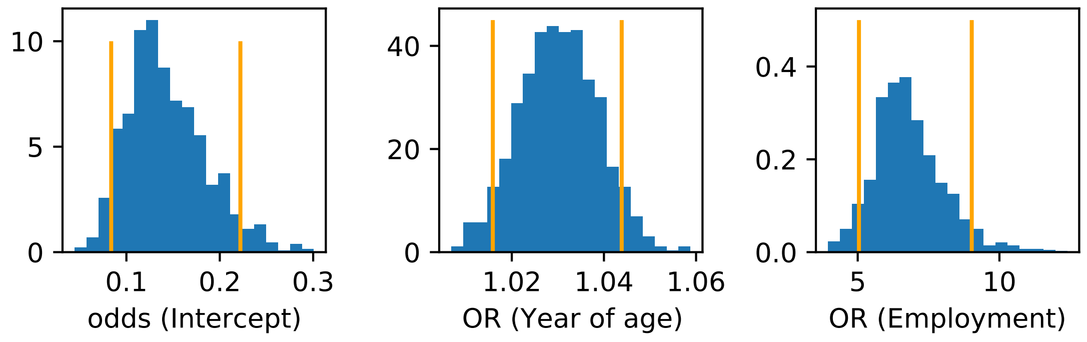


Figure 19.4: Bootstrap distributions for the baseline odds and odds ratios for age and employment in the credit scoring example.

19.3 Multiple logistic regression and confidence intervals

Principle of multiple logistic regression Just as with multiple regression, we can extend the logistic regression model (Equation 19.3) to include extra predictor variables to with corresponding coefficients:

$$P(Y = 1|X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \dots) = f(\beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \dots) \tag{19.8}$$

If the predictor variables are binary (such as employment status) we can just include them as binary variables.

Multiple logistic regression applied to credit example If we apply multiple logistic regression to the credit example, we end up with the coefficients and odds ratios shown in Table 19.1. We can see that the effect of being employed increases the odds of being awarded a loan by a factor of 6.56, an effect size of 556%. By contrast each year of age only multiplies the odds by 1.03, an effect size of 3%. To see the effect of increasing age by 10 years, we'd need to raise this OR to the power 10, and would find that the odds are only multiplied by 1.35. The effect of an increase in age from 20 to 70 is about 4.36 – still less than the effect of being in employment.

Bootstrap confidence intervals on coefficients Just as the mean and median are statistics, so are the coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ in logistic regression. We can therefore use the bootstrap to generate confidence intervals for logistic regression (Figure 19.4). The central estimate and the 95% confidence intervals computed from the 2.5% and 97.5% centiles are:

- Age: OR=1.030, CI=(1.017, 1.044)
- Employment: OR=6.562, CI=(5.110, 8.805)

19.4 Logistic regression as a classifier

Converting linear regression to a classifier Setting a threshold probability p_{thresh} corresponds to setting threshold log odds, which we'll define as c. If we choose log odds $c = 0$, this means odds of 1, i.e. the probability

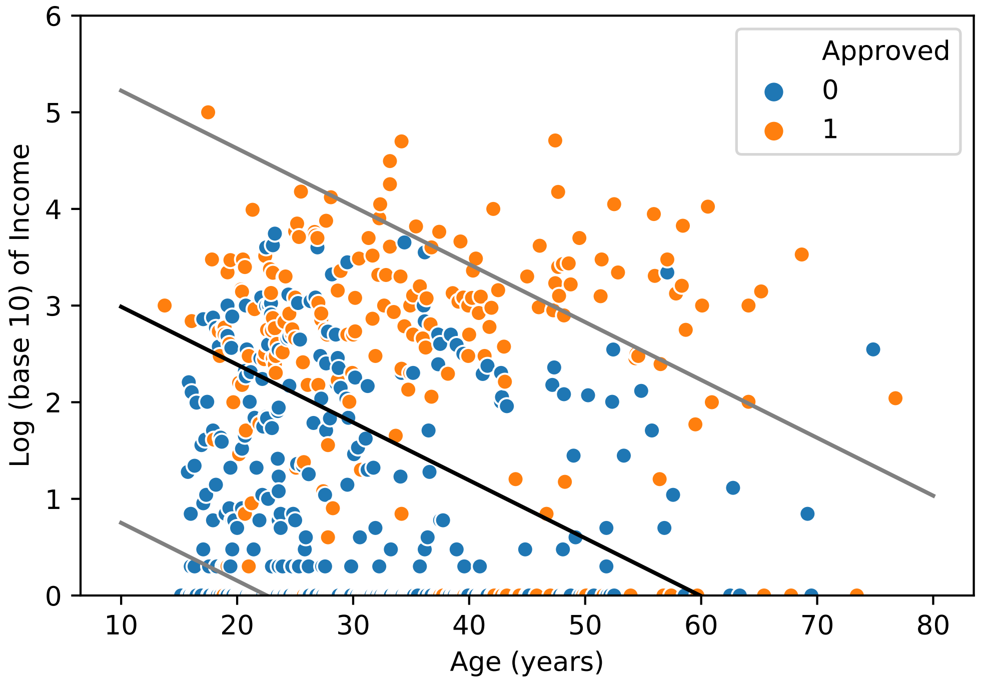


Figure 19.5: Logistic regression applied to credit approval dataset. The age and log income of each application is plotted, along with its approval status. The black line is the decision boundary found by logistic regression with an odds ratio of 1, i.e. log odds $c = 0$. The grey lines are the thresholds corresponding to odds ratios of 3 and 1/3 (i.e. 75%/25% and 25%/75%.

of success (approval) is 1/2. Substituting $c = \ln \frac{P(Y=1|x)}{1-P(Y=1|x)}$ into Equation 19.2, we find:

$$c = \beta_0 + \beta_1 x \tag{19.9}$$

This defines a linear decision boundary – in the region where $\beta_0 + \beta_1 x > c$, the log odds are greater than the threshold, and we classify unseen datapoints in this region as “Success”, and elsewhere, we classify unseen datapoints as “Failure”.

Figure 19.5 shows decision boundaries for various threshold levels when we consider two continuous variables in the credit data set: age and the log of the income. Note: as with linear regression, it often makes sense with logistic regression to transform variables so that their distribution is as normal as possible.

Transparency of logistic regression The credit agency might want to explain to its customers why their application was or was not approved. Logistic regression makes it very easy to do this, since essentially we have a credit scoring system:

- If you are in employment you score 1.625, if not you score 0
- Multiply your age by 0.029 and add the result to your score
- Round your income to the nearest 1000. Multiply the number of zeros in this figure by 0.320 and add the result to your score[5]
- If you scored more than 2.246, your credit will be approved

[5]OK, this is an approximation to a log! We could ask people to take logs or provide a table: or make the algorithm itself work in this way.

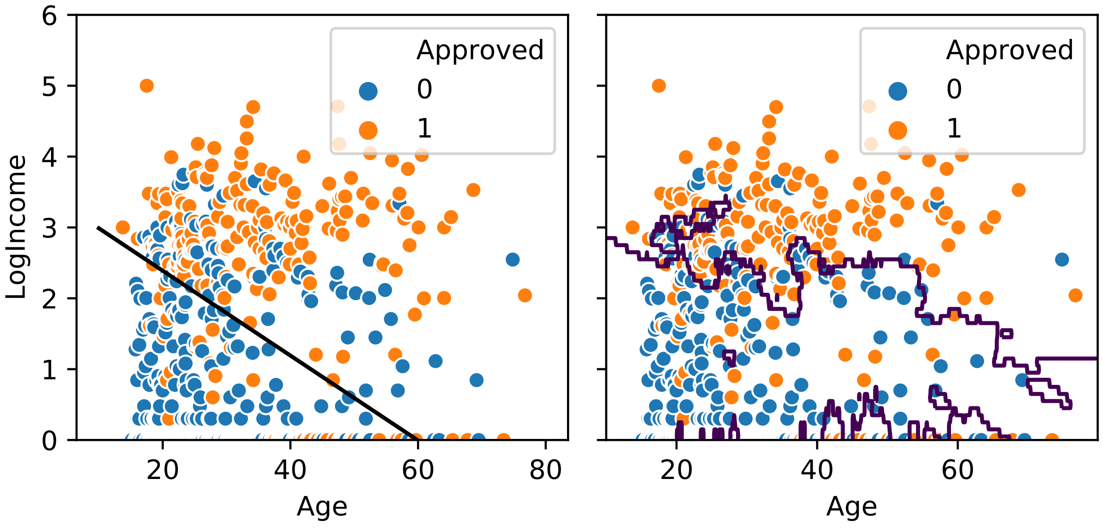


Figure 19.6: Logistic regression (left) versus 11-NN (right) applied to the credit data. The decision boundaries are shown as dark lines.

Thus, a logistic regression classifier is potentially a very **transparent** classifier. It could help to reduce the ethical harms of data science to individuals by allowing them to understand why their loan was rejected. One of the recommendations of Vallor's *Introduction to Data Ethics* is to "Promote Values of Transparency, Autonomy, and Trustworthiness" (Vallor, 2018).

Logistic regression versus *k*-nearest neighbour The logistic regression classifier differs in a number of ways from the nearest neighbour classifier:

1. The logistic regression decision boundary (obtained by setting a probability criterion) is a straight line, whereas the nearest neighbour decision boundary is nonlinear (Figure 19.6).
2. The *k*-NN thus gives more flexibility and the ability to have higher accuracy, but it is also more likely to over-fit, as seen in the chapter *k*-Nearest neighbour classification, setting hyperparameters, and metrics.
3. The logistic regression algorithm is more transparent than *k*-NN.
4. *k*-NN classifiers benefit from having standardised predictor variables as inputs; logistic regression doesn't need this, though it can help if we are regularising a logistic regression classifier (which we will not do in this course).

Often it is worth trying logistic regression first in classification problem.

19.5 Maximum likelihood estimation of logistic regression coefficients

Principle of maximum likelihood We now turn to how to estimate the logistic regression coefficients to give the best estimates $\hat{\beta}_0$, $\hat{\beta}_1$ etc. In *linear* regression we minimised the sum of squared errors between the predicted and observed response variables. We could do this analytically, ending up with a formula for the regression coefficients. In logistic regression, it doesn't make sense to minimise the sum of squared errors, since our response variable is only 0 or +1 whereas the predictor variables can have an infinite range.

Instead, we use the **principle of maximum likelihood**, which states that we adjust the model coefficients so as to maximise the likelihood that the observed data arises from the model. The resulting coefficients are referred to as the maximum likelihood estimators. Maximum likelihood can be applied to many models, not just logistic regression.

To apply the principle of maximum likelihood, we need an expression for the likelihood of all the observed data given the model, which we will do below. The likelihood is a function of β_0 and β_1. However, unlike in the case of linear regression, we can't derive formulae to give the best estimates of the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ that maximise it. We have instead to use a numerical optimisation procedure that gets us to the best estimates.

Intuition of maximum likelihood applied to logistic regression The maximum likelihood principle and derivation may look a bit complicated. We can imagine that the logistic function $P(Y = +1|X = x_i)$ is like a piece of elastic. Datapoints that are "successes" ($y_i = 1$) pull $P(Y = 1|X = x_i)$ upwards towards 1 at the location x_i, since this will make success more likely. Datapoints that are "failures" ($y_i = 0$) pull $P(Y = 1|X = x_i)$ downwards towards 0 at the location x_i. Of course, the successes and failures may be mixed up, in which case they will be competing with each other to pull the logistic function up or down.

Derivation of maximum likelihood function Assuming that all our datapoints are independent of each other, the likelihood of obtaining the full set of response variables $y_1, \dots, y_n$ is the product of the likelihood of getting each individual variable[6]. The probability of getting a success or failure is given by Equations 19.3 and 19.6. We use a trick to combine them so that the probability of success used when $y_i = 1$ and the probability of failure is used when $y_i = 0$:

$$\prod_{i=1}^{n} P(Y = y_i|X = x_i) = \prod_{i=1}^{n} (y_i f(\beta_0 + \beta_1 x_i) + (1 - y_i) f(-\beta_0 - \beta_1 x_i)) \tag{19.10}$$

Substitute $y_i = 1$ or $y_i = 0$ in the equation above to verify that each one "selects" the correct probability.

We can now use another trick. Notice that $2y_i - 1$ is equal to 1 when $y_i = 1$ and equal to -1 when $y_i = 0$. We can now express the arguments of the logistic functions in terms of $2y_i - 1$:

$$\prod_{i=1}^{n} P(Y = y_i|X = x_i) = \prod_{i=1}^{n} (y_i f((2y_i - 1)(\beta_0 + \beta_1 x_i)) + (1 - y_i) f((2y_i - 1)(\beta_0 + \beta_1 x_i)))) \tag{19.11}$$

There's now a common factor, $f((2y_i - 1)(\beta_0 + \beta_1 x_i))$, so the equation simplifies:

$$\begin{aligned}\prod_{i=1}^{n} P(Y = y_i|X = x_i) &= \prod_{i=1}^{n} (y_i + 1 - y_i) f((2y_i - 1)(\beta_0 + \beta_1 x_i)) \\ &= \prod_{i=1}^{n} f((2y_i - 1)(\beta_0 + \beta_1 x_i))\end{aligned} \tag{19.12}$$

Since the log function is a monotonically increasing function, maximising this probability is equivalent to maximising the log likelihood[7]:

$$\sum_{i=1}^{n} \ln P(Y = y_i|X = x_i) = \sum_{i=1}^{n} \ln f((2y_i - 1)(\beta_0 + \beta_1 x_i)) \tag{19.13}$$

The log of $f(u)$ is $-\ln(1 + e^{-u})$, so we can now write the optimisation equation as maximising:

$$\sum_{i=1}^{n} \ln P(Y = y_i|X = x_i) = \sum_{i=1}^{n} -\ln(1 + \exp(-(2y_i - 1)(\beta_0 + \beta_1 x_i))) \tag{19.14}$$

with respect to β_0 and β_1. Although we can compute the gradients with respect to β_0 and β_1, we can't solve the resulting equations analytically; we have to use numerical optimisation to find the best estimates $\hat{\beta}_0$ and $\hat{\beta}_1$.

[6]We will use the product notation (Greek capital letter "pi" Π) to represent a product of probabilities. For example to the probabilities of three independent events happening is $p_1 p_2 p_3$, which we represent $\prod_{i=1}^{3} p_i$.

[7]The log law $\log ab = \log a + \log b$ can be generalised using product and sum notation: $\log\left(\prod_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} \log x_i$.

Related Python Lab: Logistic regression

Possible datasets

- US death rates. Advantage: natural experiment; good for differences between groups. Not good for LR.

Chapter 20

Linear regression and inference

20.1 Inference about linear regression coefficients with the bootstrap

Recommended reading

Modern Mathematical Statistics with Applications, Chapter 12

Inference on the slope of the regression line When we first encountered linear regression, we hadn't learned about inferential statistics. It's now time to apply what we've learned about statistical inference to linear regression.

In the topic on Linear Regression we introduced the linear regression model. The response variable y depends linearly on the predictor variable x (Equation 10.1 in the topic on Linear Regression):

$$y = \beta_0 + \beta_1 x \tag{20.1}$$

The variables β_0 and β_1 are called parameters or regression coefficients and are the intercept and slope of the line respectively.

In the Linear Regression we calculated what we now know as point estimates of the intercept and slope $\hat{\beta}_0$ and $\hat{\beta}_1$ of the regression line, using the principle of least squares:

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x} \quad ; \quad \hat{\beta}_1 = \frac{\sum x_i y_i - n\overline{x}\,\overline{y}}{\sum x_i^2 - n\overline{x}^2} = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2} \tag{20.2}$$

For any set of data, we would now like to answer the following questions about the regression coefficients:

- What is our confidence in them?
- Do they represent a real effect, and not just a chance correlation in the data?
- Can we quantify uncertainty in our predictions?

Bootstrap estimation confidence intervals Since the value of the regression coefficients can be calculated from sample data, they fit the definition of a statistic. We can therefore use the bootstrap method to estimate the uncertainty in the coefficients (Figure 20.1).

As shown in the Figure, there is considerable variation in the slope – the 95% CI is (2.50, 4.24). This is evident in the sample of fits shown in the top left plot. There is also variation in the intercept – however, the scatter plot shows that it is very closely negatively correlated with the slope, since a steeper slop means that the intercept needs to be more negative to ensure that the regression lines pass through the centre of the data.

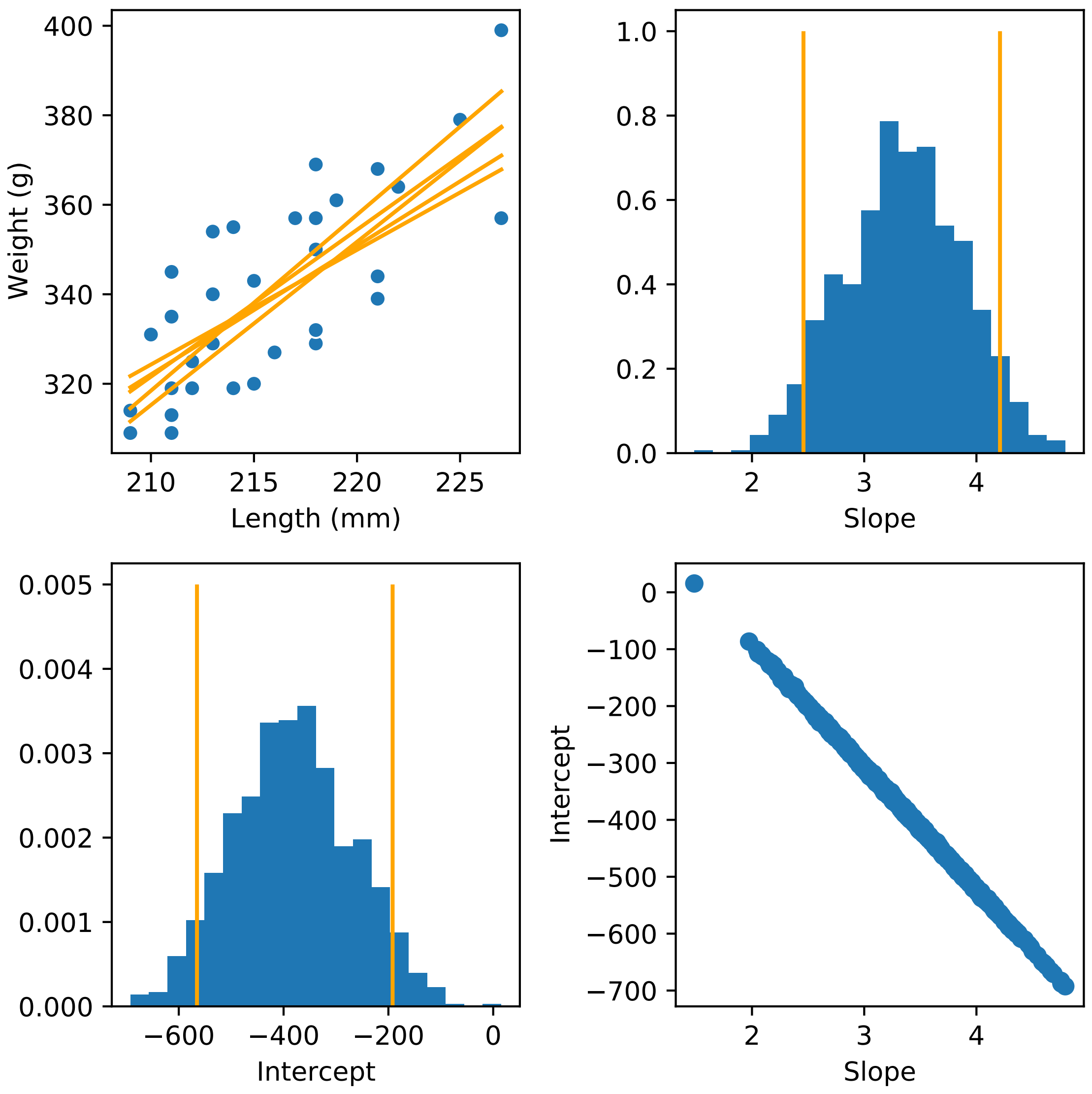


Figure 20.1: Bootstrap applied to squirrel length and weight data. Top left: the Squirrel data of Wauters and Dhondt (1989). A sample of 5 linear regression lines from the bootstrap distribution are shown. Top right: bootstrap distribution of Slope $\hat{\beta}_1$. Bottom left: bootstrap distribution of Intercept $\hat{\beta}_0$. Bottom right: scatter plot of the slope and intercept coefficients from the bootstrap simulations.

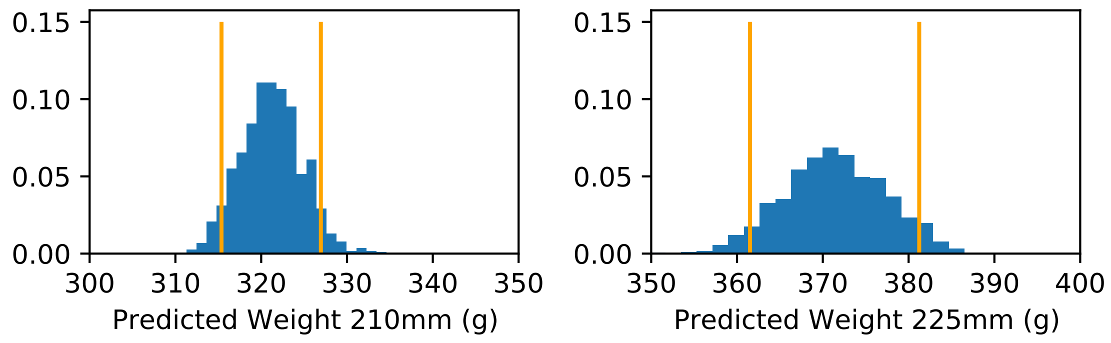


Figure 20.2: Bootstrap applied to squirrel length and weight data. Distribution of weight predictions for squirrel 210mm long (left) and 225mm long (right).

Bootstrap hypothesis testing Often we want to know if the slope of a regression model is significantly different from zero, since a non-zero slope suggests a relationship between the predictor and response variables. In other words, can we reject the null hypothesis that $\hat{\beta}_1 = 0$? The bootstrap distribution of $\hat{\beta}_1$ (Figure 20.1, top right) answers this question immediately, since the range of the bootstrap distribution does not even include 0.

Prediction uncertainty Suppose we want to predict the weight of a squirrel that weighs 225g. We can use the bootstrap distribution of coefficients to predict the weight of squirrels that are 210mm long versus those that are 225mm long (Figure 20.2). All we do is substitute in the values $x = 210$ or $x = 225$ in the equation $y = \hat{\beta}_0^* + \hat{\beta}_1^* x$ for each pair of the bootstrap estimates $(\hat{\beta}_0^*, \hat{\beta}_1^*)$. We can see that the confidence interval for squirrels of length 225mm is wider, reflecting that there are fewer data for longer squirrels than shorter ones (Figure 20.1, top left).

20.2 Sampling theory inference about linear regression coefficients

Standard error of the gradient coefficient Although the bootstrap estimates work OK, it is possible to derive algebraic expressions for the standard error of the intercept and gradient. We will first present the results. Stats packages will provide estimates of these quantities as a matter of course.

The estimated standard error in the estimator for $\hat{\beta}_1$ is:

$$\hat{\sigma}_{\hat{\beta}_1} = s_{\hat{\beta}_1} = \frac{s}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}} \tag{20.3}$$

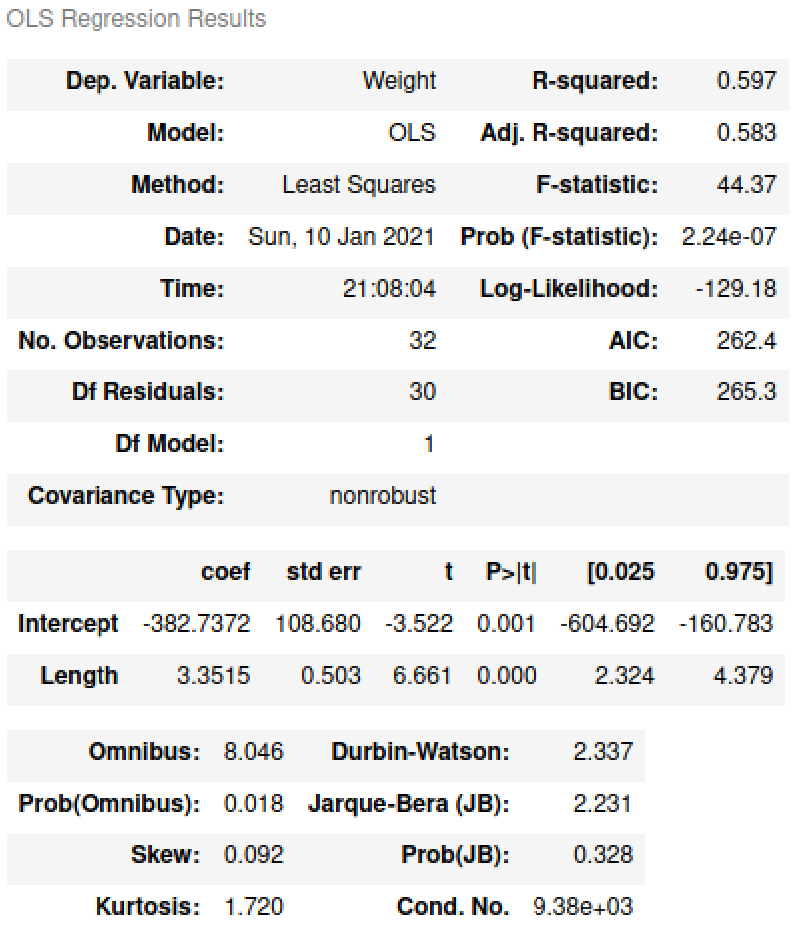
where s is the estimate for σ, the standard deviation of the residuals:

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{\text{SSE}}{n-2} \tag{20.4}$$

The more tightly the points cluster around the regression line (small s in the numerator) the smaller the standard error of the estimate of the gradient we have. Also, the larger the spread of points on the x-axis (large denominator in Equation 20.3), the smaller the standard error of the estimator.

Confidence interval for $\hat{\beta}_1$ As in our estimate of the confidence interval around the mean of small samples (Confidence intervals on the mean for small samples), we regard both $\hat{\beta}_1$ and the estimated standard error $S_{\hat{\beta}_1}$ as random variables. In order to determine the confidence intervals, we define the quantity:

$$T = \frac{\hat{\beta}_1 - \beta_1}{S_{\hat{\beta}_1}} \tag{20.5}$$



```
OLS Regression Results

   Dep. Variable:          Weight          R-squared:    0.597
           Model:             OLS      Adj. R-squared:    0.583
          Method:   Least Squares         F-statistic:    44.37
            Date: Sun, 10 Jan 2021  Prob (F-statistic): 2.24e-07
            Time:        21:08:04      Log-Likelihood:  -129.18
No. Observations:              32                 AIC:    262.4
    Df Residuals:              30                 BIC:    265.3
        Df Model:               1
 Covariance Type:       nonrobust

                coef   std err       t   P>|t|    [0.025     0.975]
Intercept  -382.7372   108.680  -3.522   0.001  -604.692   -160.783
   Length     3.3515     0.503   6.661   0.000     2.324      4.379

       Omnibus: 8.046     Durbin-Watson:    2.337
Prob(Omnibus):  0.018  Jarque-Bera (JB):    2.231
          Skew: 0.092          Prob(JB):    0.328
      Kurtosis: 1.720         Cond. No. 9.38e+03
```

Figure 20.3: Output from Python `statsmodels` applied to the squirrel dataset.

where β_1 is the true but unknown value. It turns out that this quantity has a t-distribution with $n-2$ degrees of freedom. To determine a $100(1-\alpha)\%$ confidence interval, we therefore set the probability that the variable T lies between the t critical values $\pm t_{\alpha/2,n-2}$:

$$P\left(-t_{\alpha/2,n-2} < \frac{\hat{\beta}_1 - \beta_1}{S_{\hat{\beta}_1}} < t_{\alpha/2,n-2}\right) < 1-\alpha \tag{20.6}$$

We rearrange the inside of the probability statement to find the $100(1-\alpha)\%$ confidence interval for the slope is:

$$\hat{\beta}_1 - s_{\hat{\beta}_1} t_{\alpha/2,n-2} < \beta_1 < \hat{\beta}_1 + s_{\hat{\beta}_1} t_{\alpha/2,n-2} \tag{20.7}$$

Testing hypotheses about $\hat{\beta}_1$ Often we are interested to know if the slope of a regression model is significantly different from zero – in other words, can we reject the null hypothesis that $\hat{\beta}_1 = 0$? To do this we can assume the null hypothesis by setting $\beta_1 = 0$ in Equation 20.5. Under the null hypothesis, the estimator for the slope is expected to be zero. If the estimated value of $\hat{\beta}_1$ is large relative to $S_{\hat{\beta}_1}$ then the T value will be large, and therefore far out in one of the tails of the distribution. By computing the probability mass more extreme than the T value, we get the p-value, and we can then decide if to reject or not reject the null hypothesis that the slope coefficient is different from zero.

If the p-value is large, this suggests that there may be no real relationship between the predictor and response variables, but one appears to be there by chance. Generally when the coefficient of determination R^2

is large and the number n is reasonably large, the p-value will be low, and it is reasonable to start drawing conclusions about the relationship between x and y.

In summary, we're now in a position to understand a lot more of the output produced by a stats package when it fits a linear regression model (Figure 20.3).

20.3 Derivation of standard error of estimator for slope coefficient

Where does the expression for the standard error of $\hat{\beta}_1$ (Equation 20.3) come from? First we make a subtle change to the linear regression model (Equation 20.1), by thinking of the response variable as being a random variable Y that is a sum of a deterministic linear function of the predictor variable x and a random deviation ε, which we sometimes call an **error term**:

$$Y = \beta_0 + \beta_1 x + \varepsilon \tag{20.8}$$

We will assume that ε is a random variable with an expected value of $\mathrm{E}[\varepsilon] = 0$. In principle, we can assume that is drawn from any distribution we would like, but a very common assumption is that it is drawn from a normal distribution with mean 0 and variance σ^2:

$$\varepsilon \sim N(0, \sigma^2) \tag{20.9}$$

The expression for $\hat{\beta}_1$ is now a random variable:

$$\hat{\beta}_1 = \frac{\sum (x_i - \overline{x})(Y_i - \overline{Y})}{\sum (x_i - \overline{x})^2} \tag{20.10}$$

The term $Y_i - \overline{Y}$ is exactly equal to $\varepsilon_i - \overline{\varepsilon}$, which, from the properties of ε_i has $\mathrm{E}[\varepsilon_i - \overline{\varepsilon}] = 0$ and variance $\mathrm{V}[\varepsilon_i - \overline{\varepsilon}] = \sigma^2 n/(n-1)$. Therefore, the expected variance of $\hat{\beta}_1$ is:

$$\mathrm{V}[\hat{\beta}_1] = \frac{\sum (x_i - \overline{x})^2 \mathrm{V}[\varepsilon_i - \overline{\varepsilon}]}{\sum (x_i - \overline{x})^4} = \frac{\sigma^2 n/(n-1)}{\sum (x_i - \overline{x})^2} \tag{20.11}$$

Thus the standard error of the estimator is:

$$\sigma_{\hat{\beta}_1} = \sqrt{\mathrm{V}[\hat{\beta}_1]} = \frac{\sigma \sqrt{n/(n-1)}}{\sqrt{\sum (x_i - \overline{x})^2}} \tag{20.12}$$

Since the sample variance s^2 of the residuals is an estimator of $\sigma^2 n/(n-1)$, we get the estimated standard error:

$$\hat{\sigma}_{\hat{\beta}_1} = \frac{s}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2}} \tag{20.13}$$

For more details see pp. 642–643 of Devore and Berk (2012).

20.4 Maximum likelihood estimation of linear regression coefficients

Linear regression revisited When we first encountered linear regression, to find the values of the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ we used the principle of least squares, i.e. we adjusted the values of the coefficients to minimise the sum of squared errors between the predicted and observed response variables. You might wonder why we chose the sum of squared errors rather than another measure, like the sum of the absolute differences between the data points and the regression line. One reason was that we could derive formulae for the values of $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimise the squared error. There is also a probabilistic motivation, via the principle of maximum likelihood, which we first encountered in the topic on Logistic Regression.

A probabilistic model for linear regression We've just introduced the probabilistic model of linear regression:

$$Y = \beta_0 + \beta_1 x + \varepsilon \quad ; \quad \varepsilon \sim N(0, \sigma^2) \tag{20.14}$$

If we wanted to, we could use the model to generate "fake" data points (the polite way of referring to this "fake" data is "synthetic data"). Assume that we have set the coefficients $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$. We pick a value x_i, compute $\beta_0 + \beta_1 x$ and then add a number drawn at random from a normal distribution with zero mean and variance σ^2.

However, we are not interested in generating fake data. Under the assumption that our data was indeed generated by a model of this form, and want to infer the set of parameters $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}^2$ are most consistent with data.

Principle of maximum likelihood revisited To recap from the topic on logistic regression (Maximum likelihood estimation of logistic regression coefficients), the principle of maximum likelihood states that we adjust the model coefficients so as to maximise the likelihood that the observed data arises from the model. The resulting coefficients are referred to as the maximum likelihood estimators.

To apply the principle of maximum likelihood, we need an expression for the likelihood of all the observed data given the model, which we will derive below. The likelihood is a function of β_0, β_1, and σ^2.

First, we want to compute the likelihood of one data point. The likelihood of finding a value of the predictor variable y_i at a value of the response variable x_i is:

$$P(Y = y_i|x_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}\right) \tag{20.15}$$

This is exactly the same as saying that it's the probability of drawing a point from a normal distribution with mean $\beta_0 - \beta_1 x_i$ and variance σ^2.

Derivation of maximum likelihood function Assuming that all our datapoints are independent of each other, the likelihood of the model having given rise to the full set of response variables $y_1, \dots, y_n$ given the predictor variables $x_1, \dots, x_n$ is the product of the likelihood of the model giving rise to y_i at position x_i:

$$\begin{aligned} P(y_1, \dots, y_n|x_1, \dots, x_n) &= P(Y = y_1|X = x_1) \times P(Y = y_2|X = x_2) \times \dots \times P(Y = y_n|X = x_n) \\ &= \prod_{i=1}^{n} P(Y = y_i|X = x_i) \end{aligned} \tag{20.16}$$

The notation $\prod_{i=1}^{n}$ is a short way of writing out the multiplication in the first line – it's exactly like the $\sum_{i=1}^{n}$ notation for addition.

It's convenient to work with the log likelihood. This is because the log of a product is the sum of logs of the components of the product ($\ln ab = \ln a + \ln b$). Also, the log function is a monotonically increasing function, so the values of β_1, β_2 and σ^2 that maximise the log likelihood will be exactly the same values that maximise the likelihood. Here is the log likelihood of all the data:

$$\begin{aligned} \ln P(y_1, \dots, y_n|x_1, \dots, x_n) &= \ln\left(P(Y = y_1|X = x_1) \times P(Y = y_2|X = x_2) \times \dots P(Y = y_n|X = x_n)\right) \\ &= \ln P(Y = y_1|X = x_1) + \ln P(Y = y_2|X = x_2) + \dots + \ln P(Y = y_n|X = x_n) \\ &= \sum_{i=1}^{n} \ln P(Y = y_i|X = x_i) \end{aligned} \tag{20.17}$$

We now need to find the log of the likelihood of one point (Equation 20.15). It is:

$$\begin{aligned} \ln P(Y = y_i|x_i) &= \ln\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}\right)\right) \\ &= \ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + \ln\left(\exp\left(-\frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}\right)\right) \\ &= -\ln(\sqrt{2\pi}\sigma) - \frac{(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2} \end{aligned} \tag{20.18}$$

We now substitute Equation 20.18 in Equation 20.17 to give the log likelihood of all the data:

$$\begin{aligned}\ln P(y_1,\dots,y_n|x_1,\dots,x_n) &= \sum_{i=1}^{n} \ln P(Y=y_i|X=x_i)\\ &= \sum_{i=1}^{n}\left(-\ln(\sqrt{2\pi}\sigma) - \frac{(y_i-\beta_0-\beta_1 x_i)^2}{2\sigma^2}\right)\\ &= -n\ln(\sqrt{2\pi}\sigma) - \sum_{i=1}^{n}\frac{(y_i-\beta_0-\beta_1 x_i)^2}{2\sigma^2}\end{aligned} \tag{20.19}$$

Take a moment to look at this equation. From the first linear regression topic, remember that we defined the sum of squared errors (SSE) as:

$$\mathrm{SSE} = \sum (y_i - \hat{y}_i)^2 \tag{20.20}$$

where the predicted value $\hat{y}_i = \beta_0 + \beta_1 x_i$. *The second term in Equation 20.19 is the negative of the SSE, divided by* $2\sigma^2$. Thus, to *maximise* the log likelihood, we need to *minimise* the SSE with respect to β_0 and β_1, which is exactly what we did when we derived the linear regression coefficients in semester 1. We thus have a probabilistic motivation for the principle of least squares.

The values of the coefficients We've already calculated $\hat{\beta}_0$ and $\hat{\beta}_1$ using the principle of least squares in the topic on Linear Regression:

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x} \quad ; \quad \hat{\beta}_1 = \frac{\sum x_i y_i - n\overline{x}\,\overline{y}}{\sum x_i^2 - n\overline{x}^2} = \frac{\sum (x_i-\overline{x})(y_i-\overline{y})}{\sum (x_i-\overline{x})^2} \tag{20.21}$$

However, we now have one more parameter to estimate: σ^2. To do this we maximise Equation 20.19 by differentiating it with respect to σ^2, and arrive at the maximum likelihood estimator for σ^2:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 = \frac{\mathrm{SSE}}{n} \tag{20.22}$$

Note that this is a biased estimator. An unbiased estimator, which can be obtained via estimation theory, is:

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 = \frac{\mathrm{SSE}}{n-2} \tag{20.23}$$

because are $n-2$ degrees of freedom: knowing $\hat{\beta}_0$ and $\hat{\beta}_1$ means that we only need to know $n-2$ of the y_i to work out the values of the remaining two. In practice, n is large enough that the difference between the two estimates of $\hat{\sigma}^2$ is negligible.

It is also worth noting that, since $(y_i - \hat{y}_i)$ are the residuals, that $\hat{\sigma}^2$ is the variance of the residuals.

Uncertainty estimates We can also use the maximum formulae to derive the standard error and confidence intervals for $\hat{\beta}_1$ – essentially this falls out of Equation (20.19) when we consider varying β_1.

Maximum likelihood elsewhere Maximum likelihood is a widely-used method for deriving estimators for parameters, ranging from simple models like linear regression to more complicated probabilistic machine learning ones. If it is possible to write a likelihood function for a model, it is generally possible to find its maximum.

Chapter 21

Ethical issues with supervised learning

Recommended reading

Equality law can disadvantage women in algorithmic credit decisions

Part VI

Project skills

Chapter 22

Software engineering for data science

Recommended reading

- Good enough practices in scientific computing Wilson et al. (2017)
- Reproducible Analysis Through Automated Jupyter Notebook Pipelines, Amanda Birmingham
- The scientific paper is obsolete – historical overview of notebooks

22.1 Reproducible research

What is reproducible research?

- If I have carried out an experiment or done some analysis, I can give you the instructions to re-run the experiment and analysis, and you can reproduce my results
- I can also come back to my files and run the analysis again in 3 months' time – yourself from 3 months ago doesn't answer email!

Requirements for reproducible data analysis

- Data
- Code
- Documentation

Barriers to reproducible research – data

- Data not shared
- Data lost or in incompatible format. Here are some emails I have received when requesting data the authors of scientific papers:
  - 2004 (data collected in 1990s)

    > I got these data 10 years ago, two operative systems ago, when mass storage was based on magneto-optic disks, thay I do not use any more. I will look for the potassium raw data, but I cannot promiss you to get it easily. Actually, I have to look at the Institute magazine to see if I kept the disks in a box...

  - 2021 (data collected in 2005)

> . . . I think that I do have the data somewhere but I may not have access to it at the moment. . . .

- 2021 (data collected in 1995)

> . . . Sadly the data (if it still exists) is on a dusty floppy disc somewhere – so I wouldn't be able to find or access – otherwise you would have been most welcome. . . .

Facilitators of reproducible research – data

- Journal policies
- Data repositories
  - Institutional (e.g. Edinburgh Data Share)
  - Subject-specific (e.g. EBRAINS)
  - General (e.g. Zenodo)
  - Governmental (e.g. data.gov.scot)

Barriers to reproducible research – code

- Code not shared
- Code doesn't run or takes days to get running
- Code runs but gives different results

> Although some journals now ask peer reviewers to rerun and verify code, sharing it publicly is still far from an academic norm. The amount of time researchers have to spend either helping people use their software or refuting claims stemming from its misuse is a "big worry" among many academics, says Neil Chue Hong, founding director of the Software Sustainability Institute in Edinburgh. "There are ways you can run the code that mean you won't get sensible results, but the researchers who use the code know what those ways are," he says. "It's like you or me being given a Formula One racing car, and being surprised when it crashes when it goes around the first corner." (Chawla, 2020)

- Why?
  - Missing code – not tested on clean system
  - Buggy code, maintained within a group
  - Changes in programming languages, e.g. libraries

Facilitators to reproducible research – code

- Version control and code repositories: git and github
- Unit testing: continuous integration (e.g. Travis-CI) helps to identify problems if underlying libraries get uploaded
- Conda environments, including specification of library versions used
- Virtual environments/containers
- Notebooks?

Barriers and facilitators to reproducible research – documentation Barriers:

- What do the columns in the tables mean?
- How were they collected?
- How do I run the code?

Basic solution: a README file explaining all the above!

22.2 Notebooks versus programs

Why notebooks?

Notebooks

- Mathematica (first released in 1988) was first package to have notebook interface
- Idea of combining text with computation or maths to produce a living paper, which is also reproducible
- Related to literate programming (invented by Donald Knuth author of TEX, which is written as a literate program)[1]
- The open-source Jupyter (first released under the name iPython in 2011) system[2] has implemented the notebook principle for Python, R and Julia and many other languages[3]

Advantages with notebooks

- Good for storing a one-off analysis of a few datasets, and producing figures and visualisation
- Helps reproducibility by recording list of steps in data processing
- Many packages have python interfaces – e.g. computational neuroscience packages
- Can be used on a remote server – can be helpful when data has to remain isolated (e.g. health records)

Problems with notebooks

- What if I use the same set of steps on a new dataset
  - e.g. I've run some analysis on one set of gene sequencing data - now I want to repeat it on another set?
  - Do I edit my notebook with the new dataset?
  - Or create a function?
  - We want **automation** rather than **interactivity**
- Inconsistent state [Demo]
- Collaboration on large projects

[1]Donald Knuth (the inventor of TEX, the basis for LATEX) introduced **literate programming** in 1984. The essential idea is that rather than writing in computer code, programmers embed the code in a description of the logic of the program, in human readable form. There is thus one source file (or set of source files) containing human both human-readable documentation and computer code. A preprocessor is used for two operation on these source files: (1) "tangle" – extract computer-readable source code that can be run; (2) "weave": extract human-readable documentation, with code embedded. Potential advantages of literate programming are a higher-quality program, and that the author can recall thought-processes when coming back to the program. TEX, upon which LATEX is based, is written as a literate program.

[2]Jupyter notebook is considered to be a highly influential piece of scientific software; see, e.g., *Ten computer codes that changed science* (Perkel, 2021).

[3]Although Mathematica is a closed source package, Wolfram Research have made the Wolfram engine available as a kernel for Jupyter notebooks. Thanks to 2020/21 FDS students for alerting us to this.

- Object-oriented code
- http://compbio.ucsd.edu/reproducible-analysis-automated-jupyter-notebook-pipelines/
- Version control

Best of both worlds

- If you're wanting to repeat analyses, or you're finding that you are repeating snippets between in notebooks, it might be time to create a Python module to contain the functions

22.3 Data and code management

Data versus code Many of the suggestions in this section are based on the good advice in *Good enough practices in scientific computing* (Wilson et al., 2017).

- Version control for code (e.g. using Github) is a **good thing** but does it work for data?
- VC systems deal well with "small" text files – kilobytes rather than megabytes, and definitely not gigabytes
- Thus they are good for code but not so good for data, especially large datasets
- Thus we need different solutions for storing and keeping track of changes to data and code

Data

- Save the raw data
  - Don't overwrite with a better, cleaned-up version
  - Protect, e.g. with file permissions
  - If downloading from a database, record details used to obtain data (exact query, date of retrieval, version of db on that date)
- Backup the data
- Save and share a clean version of the data
  - Cleaned
  - Open data format, e.g. CSV, JSON, YAML, XML
  - Meaningful variable names and file names
  - Metadata about meaning of columns (if not clear from original dataset)
- Make sure you the cleaned dataset is "Tidy"
  - One observation per row
  - One variable per column
  - Unit not stored with numbers – store in metadata or separate column
  - Ideally unique ID for each observation
- If generating data, share in a repository

Code

- Version control – git is currently dominant
- Documentation
- Specifying versions – this can be done using Conda environments and `.req` files

```
jupyter=1.0.0
matplotlib=2.2.3
numpy=1.15.0
pandas=0.23.4
scikit-learn=0.19.1
scipy=1.1.0
seaborn=0.9.0
python-graphviz=0.8.4
```

Jupyter notebooks and version control

- Problem: Jupyter notebooks are written in JSON, so the diffs stored in version control systems are not very intelligible
- (R markdown doesn't suffer from this problem)
- Workaround: packages such as nbdime

Chapter 23

Writing skills

Chapter 24

LaTeX and BibTeX

Related Workshop: Mid-project presentation

Appendix A

Resources

https://www.engage-csedu.org/find-resources/analyzing-airbnb-data http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/06_Dictionaries/Cancer/project8.pdf https://www.engage-csedu.org/find-resources/bmi-body-mass-index

A.1 Example visualisations and projects

http://witches.is.ed.ac.uk/

List of datasets

p.	§	Description
13	2.6.0	Titanic
13	2.6.0	Life expectancy by country
14	2.6.0	Bad form entry data
14	2.6.0	University of Edinburgh timetables
15	3.2.0	Squirrel: Wauters and Dhondt (1989). Data scraped from paper.
30	5.3.0	Life expectancy
30	5.3.0	Drinks by country
31	5.3.0	Diabetes. Categorical dep variable, mostly numeric indep variables.
34	6.2.0	Squirrel: Wauters and Dhondt (1989). Data scraped from paper.
41	7.1.0	Fruit again
56	8.3.0	Wine quality
57	9.1.0	Fruit: Iain Murray's *oranges and lemons* dataset of the widths, heights and masses of various types of fruits.
66	9.4.0	Breast cancer again
69	10.1.0	Galton's heights. From CSV in Data 8.
76	10.3.0	World population 1940–2000. Source: HYDE 3.2 database `https://dataportaal.pbl.nl/downloads/HYDE` (Klein Goldewijk et al., 2017)
78	10.3.0	A synthetic dataset that exhibits heteroscedasticity.
79	10.4.0	Diabetes again
82	11.2.0	Galton's heights
85	11.4.0	Grades data from Edge and Friedberg (1984), downloaded from website accompanying Devore and Berk (2012), where it is Example 12.25 `https://extras.springer.com/2012/978-1-4614-0390-6.zip`.
92	12.2.0	Synthetic dataset showing correlation
95	13.1.0	Scottish Index of Multiple Deprivation, 2016 edition. `https://simd.scot`
102	13.3.0	Grades data again
108	13.4.0	Breast cancer, again categorical dep variable
111	14.1.0	Squirrel again
115	14.2.0	Swain versus Alabama
120	14.4.0	Basketball salaries
120	14.4.0	Swain v Alabama
133	16.3.0	Japanese restaurant data
139	16.6.0	Basketball
142	17.1.0	Swain v. Alabama
145	17.3.0	Alameda County Jury pools. The North California branch of the American Civil Liberties Union (ACLU) investigated the numbers of Caucasian, Black/African American, Hispanic, Asian/Pacific Islander and Other people on jury panels in Alameda County.
156	18.5.0	Grades Edge and Friedberg (1984)
163	19.1.0	Credit approval dataset, from UCI.

Bibliography

Adhikari, A., DeNero, J. et al. (2020). 'Computational and inferential thinking: The foundations of data science'. Online, Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0), retrieved at various times between June 2020 and March 2021. URL `https://www.inferentialthinking.com`

Anaconda (2020). 'The state of data science 2020: Moving from hype toward maturity'. URL `https://www.anaconda.com/state-of-data-science-2020`

Anscombe, F. J. (1973). 'Graphs in statistical analysis'. *The American Statistician* **27**:17–21. URL `http://www.jstor.org/stable/2682899`

Berkson, J. (1944). 'Application of the logistic function to bio-assay'. *Journal of the American Statistical Association* **39**:357–365

Bornioli, A., Lewis-Smith, H. et al. (2020). 'Body dissatisfaction predicts the onset of depression among adolescent females and males: a prospective study.' *J Epidemiol Community Health*

Chawla, D. S. (2020). 'Critiqued coronavirus simulation gets thumbs up from code-checking efforts'. *Nature* **582**:323–324. URL `https://doi.org/10.1038/d41586-020-01685-y`

Dasu, T. and Johnson, T. (2003). *Exploratory data mining and data cleaning*. Wiley

Davies, B. (2020). 'Is web scraping legal in 2020?' URL `https://scrapediary.com/is-web-scraping-legal/`. Retrieved on 15 October 2022

Densmore, J. (2017). 'Ethics in web scraping'. URL `https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01`. Retrieved on 15 October 2022

Devore, J. and Berk, K. (2012). *Modern Mathematical Statistics with Applications*. Springer Texts in Statistics. Springer New York, New York, NY, second ed.

Edge, O. P. and Friedberg, S. H. (1984). 'Factors affecting achievement in the first course in calculus'. *Journal of Experimental Education* **52**:136–140

Gelman, A., Carlin, J. B. et al. (2004). *Bayesian Data Analysis*. Chapman & Hall/CRC, second ed.

Gelman, A. and Nolan, D. (2017). *Teaching statistics – a bag of tricks*. Oxford University Press

Hastie, T., Tibshirani, R. et al. (2009). *The elements of statistical learning*. Springer, second ed. URL `http://dx.doi.org/10.1007/b94608`

Ivory, S. B. (2021). *Becoming a critical thinker: for your university studies and beyond*. Oxford University Press

Klein Goldewijk, K., A., Beusen, J. D. et al. (2017). 'Anthropogenic land use estimates for the holocene; HYDE 3.2'. *Earth System Science Data* **9**:927–953

Kohavi, R., Henne, R. M. et al. (2007). 'Practical guide to controlled experiments on the web: Listen to your customers not to the HiPPO'. In P. Berkhin, R. Caruana, X. Wu and S. Gaffney, eds., *KDD-2007 Proceedings of the thirteenth ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 959–967. Association of Computing Machinery, New York, USA

Kramer, A. D., Guillory, J. E. et al. (2014). 'Experimental evidence of massive-scale emotional contagion through social networks.' *Proc Natl Acad Sci U S A* **111**:8788–90. URL `https:dx.doi.org:/10.1073/pnas.1320040111`

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press, sixth ed. URL `http://www.inference.phy.cam.ac.uk/itprnn/book.pdf`

Murray, I. (2006). 'Oranges, lemons and apples dataset'. URL `http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/`

Perkel, J. M. (2021). 'Ten computer codes that transformed science'. *Nature* **589**:344–348. URL `https://doi.org/10.1038/d41586-021-00075-2`

Rutherford, A. (2022). *Control: the Dark History and Troubling Present of Eugenics*. Weidenfeld & Nicolson, London

Schwabish, J. (2021). *Better data visualizations: a guide for scholars, researchers and wonks*. Colombia University Press

Scottish Government (2016). 'Scottish index of multiple deprivation (SIMD) 2016'. URL `https://www.webarchive.org.uk/wayback/archive/20200117165925/https://www2.gov.scot/SIMD`

Tufte, E. (1982). *The visual display of quantitative information*. Graphics Press, Cheshire, Connecticut

Tufte, E. (2001). *The visual display of quantitative information*. Graphics Press, Cheshire, Connecticut, second ed. 1st ed was in 1982

Turing Way Community, T. (2022). *The Turing Way: A Handbook for Reproducible Data Science (Version v1.0.3)*. Zenodo. URL `http://doi.org/10.5281/zenodo.6909298`

Vallor, S. (2018). 'An introduction to data ethics'. Online. URL `https://www.scu.edu/media/ethics-center/technology-ethics/IntroToDataEthics.pdf`

Verhulst, P.-F. (1845). 'Recherches mathématiques sur la loi d'accroissement de la population'. *Nouveaux mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles* **18**:4–55. URL `https://gdz.sub.uni-goettingen.de/id/PPN129323640_0018?tify={%22pages%22:[14],%22panX%22:0.459,%22panY%22:0.815,%22view%22:%22info%22,%22zoom%22:0.721}`

Verma, I. M. (2014). 'Editorial expression of concern: Experimental evidence of massive-scale emotional contagion through social networks.' *Proc Natl Acad Sci U S A* **111**:10779. URL `https://dx.doi.org/10.1073/pnas.1412469111`

Vopson, M. M. (2021). 'The world's data explained: how much we're producing and where it's stored'. *The Conversation* URL `https://theconversation.com/the-worlds-data-explained-how-much-were-producing-and-where-its-all-stored-159964`

Wasserstein, R. L. and Lazar, N. A. (2016). 'The ASA statement on *p*-values: Context, process, and purpose'. *The American Statistician* **70**:129–133. URL `https://doi.org/10.1080/00031305.2016.1154108`

Wauters, L. A. and Dhondt, A. A. (1989). 'Variation in length and body weight of the red squirrel (*Sciurus vulgaris*) in two different habitats'. *Journal of Zoology* **217**:93–106

Wexler, S., Shaffer, J. et al. (2017). *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. Wiley

Wickham, H. (2014). 'Tidy data'. *Journal of Statistical Software* **59**. URL `https://www.jstatsoft.org/index.php/jss/article/view/v059i10/v59i10.pdf`

Wilson, G., Bryan, J. et al. (2017). 'Good enough practices in scientific computing'. *PLOS Computational Biology* **13**:1–20. URL `https://doi.org/10.1371/journal.pcbi.1005510`

Xu, D. and Tian, Y. (2015). 'A comprehensive survey of clustering algorithms'. *Annals of Data Science* **2**:165–193. URL `https://doi.org/10.1007/s40745-015-0040-1`

Yanai, I. and Lercher, M. (2020). 'A hypothesis is a liability'. *Genome Biol* **21**:231. URL `https://doi.org/10.1186/s13059-020-02133-w`

Index