University of Manitoba

Mathematics of Data Science

Julien Arino

Department of Mathematics

Fall 2023
Version of September 5, 2023

Contents

1 Data Science and Mathematics? 7
- 1.1 What is Data Science? . . . 7
  - 1.1.1 The data deluge . . . 8
- 1.2 Why bother about mathematics? . . . 8
- 1.3 Where to go for more information . . . 8
- 1.4 Remark about this document . . . 8

2 `R` and data 11
- 2.1 Introduction to R and jupyter notebook . . . 11
- 2.2 Grabing the Canadian census data and putting it into shape . . . 11

3 Least squares problems 21
- 3.1 From interpolation to fitting . . . 21
- 3.2 Least squares problems . . . 24
- 3.3 Solving by brute force using a genetic algorithm . . . 27
- 3.4 How about a little finesse? . . . 28
- 3.5 Fitting something more complicated . . . 31
  - 3.5.1 Fitting the quadratic . . . 32
  - 3.5.2 Fitting the exponential . . . 32

4 Matrix factorisations 35
- 4.1 Orthogonal matrices . . . 35
- 4.2 The Gram-Schmidt orthonormalisation procedure . . . 37
  - 4.2.1 Projections onto subspaces . . . 38
  - 4.2.2 The Gram-Schmidt process . . . 38
- 4.3 The QR factorisation . . . 38
  - 4.3.1 Back to least squares . . . 38
- 4.4 The singular values decomposition (SVD) . . . 39
  - 4.4.1 Computing the SVD (case of $\neq$ eigenvalues) . . . 40
  - 4.4.2 Computing the SVD (case of $\neq$ eigenvalues) . . . 41
  - 4.4.3 Computing the SVD (case where some eigenvalues are $=$) . . . 41
- 4.5 Compressing images . . . 42
  - 4.5.1 Doing things "by hand" . . . 44

4.5.2 Doing things using proper functions . . . 44

5 Principal component analysis (PCA) 47
5.1 Brief "review" of some probability concepts . . . 47
5.1.1 Hockey players (eh!) . . . 54

6 Graph theory ... theory 63
6.1 Introduction and preliminaries . . . 63
6.1.1 Graphs versus networks . . . 63
6.1.2 Graphs vs digraphs vs multigraphs vs multidigraphs vs 63
6.1.3 The bridges of Königsberg . . . 64
6.1.4 Finding a cycle with all vertices . . . 64
6.1.5 How far is it to drive through n cities? . . . 64
6.2 Binary relations . . . 65
6.3 Undirected graphs . . . 66
6.3.1 Undirected graph . . . 66
6.3.2 Order and size of graph . . . 67
6.3.3 Relationships between vertices and edges, nature of the edges . . . 68
6.3.4 Degree of a vertex . . . 69
6.3.5 Regular, complete, bipartite and other notable graphs . . . 70
6.3.6 Isomorphic graphs . . . 73
6.3.7 Subgraphs, unions of graphs . . . 74
6.3.8 Walks, trails, paths . . . 75
6.3.9 Eulerian graphs . . . 76
6.3.10 Hamiltonian graphs . . . 77
6.3.11 Connectedness . . . 78
6.3.12 Planar graphs . . . 82
6.4 Directed graphs . . . 88
6.4.1 Directed graph . . . 88
6.4.2 Degrees in digraphs . . . 90
6.4.3 Walks, paths, etc. . . . 91
6.4.4 Connectivity in digraphs . . . 92
6.4.5 Orientable graphs . . . 94
6.5 Trees . . . 94
6.6 Matrices associated to a graph/digraph . . . 100
6.6.1 Adjacency matrices . . . 100
6.6.2 Other matrices associated to a graph/digraph . . . 102
6.6.3 Linking graphs and linear algebra . . . 104

7 Quantifying graphs 107
7.1 Measures specific to vertices . . . 108
7.1.1 Centre of a graph . . . 108
7.1.2 Centrality – Betweenness and closeness . . . 109

7.1.3 Periphery of a graph . . . 111
7.1.4 Degree distribution . . . 112
7.2 Measures at the graph level . . . 114
7.2.1 Circumference & Girth . . . 114
7.2.2 Graph density . . . 115
7.2.3 Graph connectivity . . . 116
7.2.4 Cliques . . . 118
7.2.5 k-cores . . . 118

8 The PageRank algorithm 121
8.1 Markov chains . . . 121
8.2 Running example – Mendelian inheritance . . . 122
8.3 Repetition of the process . . . 124
8.4 Regular Markov chains . . . 126
8.5 Back to the genetics example . . . 127
8.6 Changing the setting of the genetic experiment . . . 128
8.7 Absorbing Markov chains . . . 129

A Review/presentation of some required concepts 133
A.1 Sets . . . 133
A.1.1 Sets and elements . . . 133
A.1.2 Quantifiers . . . 134
A.1.3 Intersection and union of sets . . . 134
A.2 Just enough logic to get by . . . 135
A.3 Vectors and vector spaces . . . 136
A.3.1 Vectors . . . 136
A.3.2 Vector space . . . 136
A.3.3 Norms . . . 136
A.3.4 Standard basis vectors . . . 137
A.3.5 Dot product . . . 137
A.3.6 Some results stemming from the dot product . . . 137
A.3.7 Scalar and vector projections . . . 137
A.4 Complex numbers . . . 138
A.4.1 Solving quadratic equations . . . 139
A.4.2 Why this matters . . . 140
A.5 Linear systems and matrices . . . 141
A.5.1 Linear systems . . . 141
A.6 Matrix arithmetic . . . 142
A.6.1 Symmetric matrices . . . 143
A.6.2 Determinants . . . 145
A.7 Diagonalisation . . . 147
A.7.1 Eigenvalues / Eigenvectors / Eigenpairs . . . 147
A.7.2 Diagonalisation . . . 147

A.8 Linear independence/Bases/Dimension 148
A.9 Linear algebra in a nutshell 149

Chapter 1

What is Data Science and why should you care about mathematics as a data scientist?

1.1 What is Data Science?

"Nobody knows" is probably quite an adequate answer to this question. In days of yore (circa 2010), people talked about Big Data, prompting the following declaration, attributed to Dan Ariely (Duke University):

> Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...

The vocabulary has evolved, the term *big data* became *complex data* and more recently, people have been talking about data science. According to Wikipedia (emphasis mine)

> Data science is an **interdisciplinary field** that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from **structured** and **unstructured data**, and apply knowledge and actionable insights from data across a broad range of application domains.
>
> [..] It uses techniques and theories drawn from many fields within the context of **mathematics**, **statistics**, **computer science**, **information science**, and **domain knowledge**.

Data science is nothing new (some statisticians argue it is just another name for statistics), but it has become prominent in recent years as a consequence of the unprecedented mass of information generated and collected by our modern societies.

1.1.1 The data deluge

One speaks of *information explosion* or *data deluge*. See some considerations, e.g., here.

A wide variety of jobs

We have absolutely insane amounts of data and we try to make sense of it. However, except for the more or less fixation of the name, the situation has not improved significantly since the days of Ariely's quote: data science is a hodge-podge that contains everything but the kitchen sink.

To caricature - two main types of data: structured and unstructured - two main branches: statistics and computer science - two main types of jobs: users and developpers

1.2 Why bother about mathematics?

Recall that I said there were two main branches, statistics and computer science, and two main types of jobs: users and developpers. So why a course on Math of Data Science?

In short, if you plan to be a user and are not curious about the *how* and the *why* and can tolerate some errors due to misuse of methods, then you probably do not care about this course.

In other cases, many of the concepts used have their roots in math and to understand where the methods are coming from and, even more importantly, to develop new methods, math is often required

Warning!

We will barely brush the surface here:

- Some techniques from linear algebra - Some graph theory ideas

There is a lot more to see!!!

1.3 Where to go for more information

The Faculty of Science at the University of Manitoba has created the Data Science NEXUS (link).

- Education component - Data Science Undergraduate Program (Fall 2021) - Data Science Master's Program - Master of Business Analytics

- Training (workshops, COOP, internships, etc.)

- Events (conferences, etc.)

1.4 Remark about this document

This document is written in `RSweave`, which is very similar to `Rmarkdown`, but allows to write in native LaTeX. All figures and most data processing is run by `R` at the time of compilation. `R` code is also displayed frequently, when it is novel, in the sense that

it has not been shown before. Otherwise, it is hidden. For example, the code used to generate the first figure is shown; in subsequent figures, code is only shown if additional options are used, etc.

Chapter 2

Getting started – Learning R and collecting data

2.1 Introduction to R and jupyter notebook

In order to illustrate the use of Jupyter Notebook and R, let us prepare for the next lecture using a notebook.

Notebooks run online (here on syzygy.ca). It uses two (mainly) types of "cells". A cell like this one is a text cell. Text is formatted using 'markdown', which is a simple text description language yet has relatively powerful capabilities. See here, for instance, for details.

2.2 Grabing the Canadian census data and putting it into shape

The first steps in analysing data typically involve acquiring it, then putting it into a form that is appropriate for the analysis steps that will follow. The latter step is often called **data wrangling**.

To illustrate the process, let us consider the evolution of the population of Canada through time. For this, we want to find census data. We search the web for "canada historical census data csv", since `csv` (comma separated values) files are very easy to use with `R`. Our search returns a website, here, with a `csv` for 1851 to 1976. We follow the link to Table A2-14, where we find another link (this one), this time to a `csv` file. This is what we use below.

The first step is to read in the data. The function `read.csv` reads in a file, potentially directly from the web. We assign the result to the variable `data_old`. The reason for using the suffix *old* should become clear soon. Note that in order for the link to appear clearly in this text, we first store it in two variable, one for the base url and one for the specific page; this is not a required step by any means. The way we do this, however, is of interest, as it is a process that is often used. Once the two strings are

stored, we paste them one to another using the function `paste0`, which appends all its arguments as a string without adding any space (there is also a function `paste` that does the same thing but with separating spaces).

```
> url_base = "https://www150.statcan.gc.ca/"
> url_page = "n1/en/pub/11-516-x/sectiona/A2_14-eng.csv?st=L7vSnqio"
> url_data = paste0(url_base, url_page)
> data_old = read.csv(url_data)
```

When this is done, it is always a good idea to take a look at the first few rows in the resulting table, to see what things look like. The function `head` shows the first few lines in the argument, eight by default. The call to `knitr::kable` allows to use the function `kable` from the `knitr` library without loading the whole library. The function `kable` formats a table for display in a much better way than `R` does by default.

```
> knitr::kable(head(data_old))
```

```
|  X|Series.A2.14. |Population.of.Canada..by.province..census.dates..1851.to.1976 | X.
|--:|:-------------|:-----------------------------------------------------------|---
| NA|              |                                                            |  N
| NA|Year          |Canada                                                      |  N
| NA|              |                                                            |  N
| NA|              |                                                            |  N
| NA|              |2                                                           |  N
| NA|              |                                                            |  N
```

We use `kable` quite often, actually, so let us load the library (I am assuming it is installed and not using the "safe" way of loading a library explained earlier).

```
> library(knitr)
```

Returning to the data, obviously, this does not make a lot of sense. Take a look at the first few lines in the `csv` file. (This is something you would usually do even before loading the data into `R`.) They take the form

```
,Series A2-14.,"Population of Canada, by province, census dates, 1851 to 1976",,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,
,Year,Canada,,Newfound-,Prince,,Nova,New,Quebec,Ontario, Manitoba,,Saskat-,,Alberta,,British,,
,,,,land,Edward,,Scotia,Brunswick,,,,,chewan,,,,Columbia,,Territory,Territories,,
,,,,,Island,,,,,,,,,,,,,,,,
,,2,,3,4,,5,6,7,8,9,,10,,11,,12,,13,14,,
,,,,,,,,,,,,,,,,,,,,,,
```

This is frequent and results from good data storage practice: the first row (and sometimes quite a few more) contains metadata that helps understand what is in the table. However, in the present case, this is not something we care about because we know where the data is coming from, so we can discard this row. The function `read.csv` takes the optional argument `skip=`, which indicates how many lines to skip at the beginning. The second line is also empty, so let us skip it too.

```
> data_old = read.csv(url_data, skip = 2)
```

This gives a slightly better looking table.

```
|  X|Year |Canada     | X.1|Newfound. |Prince  | X.2|Nova    |New       |Quebec    |Ontario
|--:|:----|:----------|---:|:---------|:-------|---:|:-------|:---------|:---------|:---------
| NA|     |           |  NA|land      |Edward  |  NA|Scotia  |Brunswick |          |
| NA|     |           |  NA|          |Island  |  NA|        |          |          |
| NA|     |2          |  NA|3         |4       |  NA|5       |6         |7         |8
| NA|     |           |  NA|          |        |  NA|        |          |          |
| NA|1976 |22,992,604 |  NA|557,725   |118,229 |  NA|828,571 |677,250   |6,234,445 |8,264,465
| NA|     |           |  NA|          |        |  NA|        |          |          |
```

Here, there is the further issue that to make things legible, the table authors used 3 rows (from 2 to 4) to encode for long names (e.g., Prince Edward Island is written over 3 rows). Because we are only interested in the total population of the country and the year, let us get rid of the first 4 rows and of all columns except the second (`Year`) and third (`Canada`). Let us take a look at what we now have, both at the top of the table (using `head()`):

	Year	Canada
5	1976	22,992,604
6		
7	1971	21,568,311
8	1966	20,014,880
9	1961	18,238,247
10	1956	16,080,791

and at the end of the table (using `tail()`):

	Year
24	
25	Includes 485 members of the Royal Canadian Navy whose province of residence is not known.
26	Included with Northwest Territories.
27	For the discussion of the ambiguities and under-enumeration contained in these figures consult the notes
28	see notes to series A15-53.
29	1848 figure.

There are still quite a few remaining issues:

1. there are some empty rows;
2. the last few rows need to be removed too, they contain remarks about the data;
3. the population counts contain commas;
4. it would be better if years were increasing.

Let us fix these issues.

- For 1 and 2, this is easy: remark that the `Canada` column is empty for both issues. Remark as well (if viewing the data from within RStudio or Jupyter Notebook) that below `Canada` (and `Year`, for that matter), the text *<chr>*. This means that entries in the column are *characters*. Looking for empty content therefore means looking for empty strings. So we want to keep the rows where `Canada` does not equal the empty string.

  ```
  > data_old = data_old[which(data_old$Canada != ""),]
  ```

- To get rid of commas, we just need to substitute an empty chain for ",".

  ```
  > data_old$Canada = gsub(",", "", data_old$Canada)
  ```

- To sort, we find the order for the years and apply it to the entire table.

  ```
  > order_data = order(data_old$Year)
  > data_old = data_old[order_data,]
  ```

- Finally, as remarked above, both the year and the population are strings. This means that in order to plot anything, we will have to indicate that these are numbers, not strings.

  ```
  > data_old$Year = as.numeric(data_old$Year)
  > data_old$Canada = as.numeric(data_old$Canada)
  ```

Let us see what the table looks like now.

	Year	Canada
23	1851	2436297
22	1861	3229633
21	1871	3689257
20	1881	4324810
19	1891	4833239
17	1901	5371315

Row numbers are a little weird (the indicate the number of the row in the table as we originally loaded it), so let us fix this. Note that this is purely cosmetic.

```
> row.names(data_old) = 1:dim(data_old)[1]
```

So we have

Year	Canada
1851	2436297
1861	3229633
1871	3689257
1881	4324810
1891	4833239
1901	5371315
1911	7206643
1921	8787949
1931	10376786
1941	11506655
1951	14009429
1956	16080791
1961	18238247
1966	20014880
1971	21568311
1976	22992604

Note that the row numbers are not show any more: they "make sense" and thus `kable` has "decided" not to show them. This is looking quite good. However, this data only goes to 1976. That is close to 50 years missing, surely there must be more recent data!

Looking around, we find another table here. There is a download csv link on that page (this one), let us see where this leads us. Note that the link is very long so I am not showing the `read.csv` command used. Refer to the `Jupyter Notebook` or `Rmarkdown` files on the GitHub repository to see it. The table is 720KB, so there must be more to this than just the population. To get a sense of that, we could dump the whole data frame. Again, this is too much for lecture notes, refer to the worksheets for details. Let us, nonetheless, take a quick peek inside the file. For this, instead of using `head()`, let us pick 10 rows at random.

```
> idx = sort(sample(1:dim(data_new)[1], 10))
```

giving

	GEOGRAPHY.NAME	CHARACTERISTIC	YEAR.S.
395	Newfoundland and Labrador	Population by single years of age: 18 years	2011
701	St. John's	Population by single years of age: 68 years	2011
810	Prince Edward Island	Total private dwellings occupied by usual residents	2011
907	Prince Edward Island	Population by single years of age: 75 years	2011
1004	Prince Edward Island	% of married-couple families	1966
1697	New Brunswick	% of two-person households	2011
1727	New Brunswick	% of five-or-more-person households	1996
1854	Moncton	Population by single years of age: 38 years	2011
1914	Moncton	Population by single years of age: 98 years	2011
2467	Saguenay	Population by single years of age: 13 years	2011

This is a very different way of storing the information from the first table, which had, to caricature, columns with the different regions of interest and rows the years. Here, different contain different information. To get a better sense of what we are looking at, let us first look at the columns.

```
> colnames(data_new)
```

```
[1] "GEOGRAPHY.NAME" "CHARACTERISTIC" "YEAR.S."        "TOTAL"
[5] "FLAG_TOTAL"
```

This is a different way of thinking about the data. `GEOGRAPHY.NAME` and `CHARACTERISTIC`, taken as a pair, define an attribute, whose `TOTAL` (value) is then documented for each `YEAR.S` in the table. The column `FLAG_TOTAL` is used to comment on individual data points. By selecting the relevant *geography* and *characteristic*, we therefore get the time series we want. To find what value of *geography* and *characteristic* to use, we parse through the output of, say,

```
unique(data_new$GEOGRAPHY.NAME)
unique(data_new$CHARACTERISTIC)
```

that return the list of distinct (unique) values in both these columns. The output is not shown here because these commands generate a substantial number of results: there are 12 unique geographies and 184 unique characteristics in the table. Looking through these results, we find that the information we want has geography "Canada" and characteristic "Population (in thousands)". In passing, from this we also remark that the population of Canada is expressed in thousands, so once we have selected what we want, we will need to multiply the value by 1,000 in order to have data in the same units as in `data_old`.

There are many ways to select rows. Let us present three different mechanisms.

The basic (and reliable) way Let us proceed as follows: we want the rows where the geography is "Canada" and the characteristic is "Population (in thousands)". Let us find the indices of rows that satisfy the first criterion, those that satisfy the second; if we then intersect these two sets of indices, we have the indices of the rows we are interested in.

```
> idx_CAN = which(data_new$GEOGRAPHY.NAME == "Canada")
> idx_char = which(data_new$CHARACTERISTIC == "Population (in thousands)")
> idx_keep = intersect(idx_CAN, idx_char)
```

Once these indices have been selected, we can retain only those rows in `data_new` that are of interest to us.

```
> data_new = data_new[idx_keep,]
```

Thinking SQL If you have any experience with SQL, a language for maintaining and querying databases, it can be tempting to solve this selection problem using SQL syntax. This is made possible by using the library `sqldf`, which allows to perform SQL queries on data frames.

```
> if (!require("sqldf")) {
+   install.packages("sqldf")
+ }
> query = "SELECT * FROM data_new
+ WHERE `GEOGRAPHY.NAME`='Canada'
+ AND `CHARACTERISTIC`='Population (in thousands)'"
> data_new = sqldf(query)
```

The modern way: `dplyr` `dplyr` is part of the `tidyverse`, an ecosystem of `R` libraries meant to facilitate data manipulation. Personnally, I tend to be more familiar with SQL, but there are similarities. `dplyr` also makes use of the pipe `%>%`, which is extremely powerful when manipulated correctly. (This is the same idea as the unix pipe |.)

Back to our wrangling We want to concatenate this data frame with the one from earlier. To do this, we need the two data frames to have the same number of columns and, actually, the same column names and entry types (notice that `YEAR.S.` in `data_new` is a column of characters). So what remains to do:

1. Rename the columns in the old data (`data_old`) to `year` and `population` (personally, I prefer lowercase column names).

   ```
   > colnames(data_old) = c("year", "population")
   ```

2. Keep only the relevant columns in `data_new` and rename them to match column names in `data_old`.

```
> data_new = data_new[,c("YEAR.S.","TOTAL")]
> colnames(data_new) = c("year", "population")
```

3. Transform year in `data_new` to numbers.

   ```
   > data_new$year = as.numeric(data_new$year)
   ```

4. Multiply population by 1,000 in `data_new`.

   ```
   > data_new$population = data_new$population*1000
   ```

5. We already have data up to and including 1976 in `data_old`, so get rid of that in `data_new`.

   ```
   > data_new = data_new[which(data_new$year>1976),]
   ```

6. Append the rows of `data_new` to those of `data_old`. To simplify later use, we call `data` this aggregated data.

   ```
   > data = rbind(data_old,data_new)
   ```

Let us see what this looks like when all of this is done; see Figure 2.1. For information, the command used to generate Figure 2.1 is the following:

```
plot(data$year, data$population,
     type = "b",
     lwd = 2,
     xlab = "Year",
     ylab = "Population")
```

In case we need the data elsewhere, we save it.

```
> write.csv(data, file = "Canada_census.csv")
```

Saving processed data is usually a good idea. The preprocessing steps we have just carried out can be quite costly computationally in the case of a large data set. Also, the source of the data may move: websites, even institutional ones like Open Data portals, have a tendency to rename files occasionally. Some files come with permanent identifiers that in principle make them impervious to these changes, but this is not yet the norm.

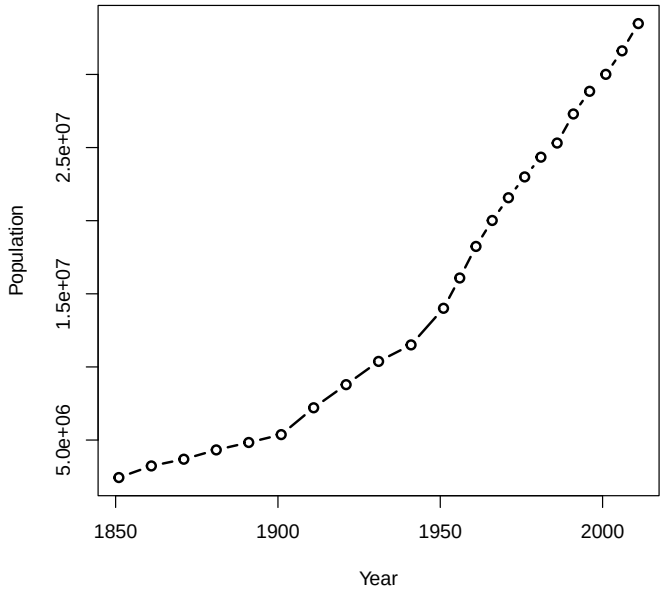


Figure 2.1: Evolution of the population of Canada.

Chapter 3

Least squares problems

In this chapter, we consider several matrix methods useful in Data Science applications, with different purposes in mind. Firstly, explain the principles; secondly, understand why a method works; thirdly, show how to use the method in practice, both "by hand" and using computational software.

We present some matrix theory, focusing on the specific results needed for the topics under consideration. Other required material is found in Appendix A.

To motivate least squares problems, remember the data we gathered in Chapter 2 about the evolution of the population of Canada through time. As we saved that data then, let us load it

```
> data = read.csv("Canada_census.csv")
```

and plot it, with the result shown in Figure 3. A public official, in view of this figure, might be interested in getting a sense of what the population of Canada could be expected to be in, say, 2050. There are several ways to do that. One is to use so-called linear least squares (often called least squares for short), which is what we study here.

3.1 From interpolation to fitting

Let us start with the simplest possible case: if we have just two points, then obviously, it is easy to find the parameters a_0 and a_1 such that the line

$$\ell : \ y = a_0 + a_1 x \tag{3.1}$$

goes through the two points. This is called a problem of **interpolation**. There are several ways to do that, but let us do so in a way that will be an inspiration for later problems.

Suppose the two points are (x_1, y_1) and (x_2, y_2). Since they are on the line ℓ, they must satisfy the equation (3.1), and thus

$$\begin{aligned} y_1 &= a_0 + a_1 x_1 \\ y_2 &= a_0 + a_1 x_2. \end{aligned}$$

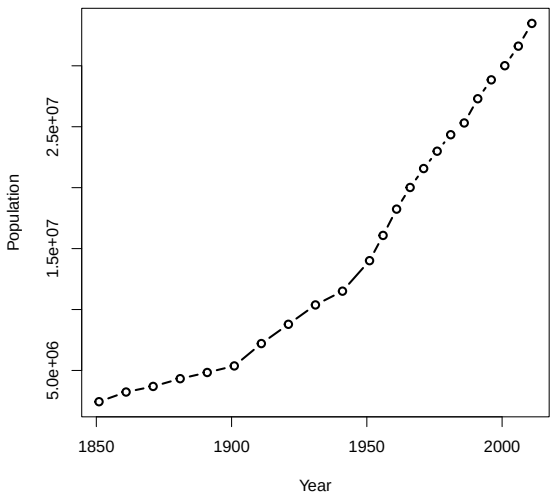


Figure 3.1: Evolution of the population of Canada.

Recall that what we seek are the coefficients a_0, a_1 of ℓ, so we write this as a linear system with unknowns a_0 and a_1:

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},$$

which we write $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \text{ and } \mathbf{b} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

From Theorem A.62, we know that this system has the unique solution $\mathbf{x} = A^{-1}\mathbf{b}$ if and only if $\det(A) \neq 0$. We easily obtain that $\det(A) = x_2 - x_1$ and thus $\det(A) \neq 0$ (and we have a unique solution) if and only if $x_1 \neq x_2$. This makes sense: if $x_1 = x_2$, this means ℓ is vertical, which cannot be described by an equation such as (3.1).

Let us take a quick look at a numerical example. Suppose we have two points $(1, 3)$ and $(3, 5)$. In `R`, we can for instance create a list and store the values there.

```
> points = list()
> points$x = c(1,2)
> points$y = c(3,5)
```

Recall that matrix inversion in `R` is through the function `solve`, which, when provided with a second (vector) argument, actually computes $A^{-1}\mathbf{b}$, so

```
> A = matrix(c(1,points$x[1],
+              1,points$x[2]),
+            nr = 2, byrow = TRUE)
> coefs = solve(A, points$y)
```

To plot the line with the coefficients we just obtained, we use the function `abline`, which is quite versatile. With an argument `coef=`, it plots the line with these coefficients. So, here, we will add the command `abline(coef = coefs, lwd = 2, col = "red")` after the `plot` command, giving Figure 3.1.

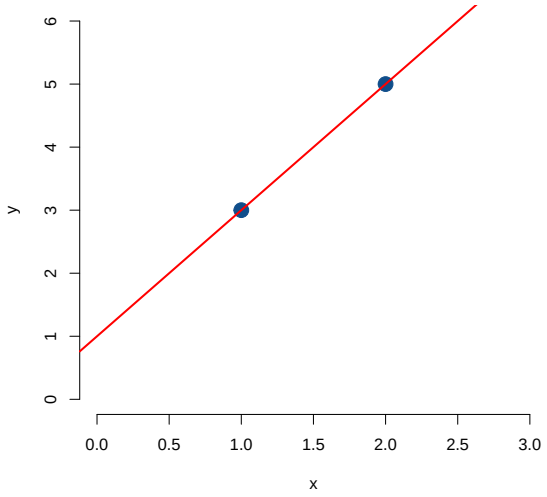


Figure 3.2: Two lonely points and an interpolating line.

However, we rarely have just two points and that is where things get more complicated. Suppose we add a third point, say, $(3, 4)$, to the previous two. So we now have the situation shown in Figure **??** and, clearly, since the points are not colinear, it is impossible to find coefficients a_0, a_1 of a curve $y = a_0 + a_1 x$ that goes through the three points.

Mathematically, this is also obvious. Reason as we did before: we have, now, three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) that must all satisfy the equation (3.1), i.e.,

$$\begin{aligned} y_1 &= a_0 + a_1 x_1 \\ y_2 &= a_0 + a_1 x_2 \\ y_3 &= a_0 + a_1 x_3. \end{aligned}$$

As we did earlier, we write this in the form of a linear system $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \text{ and } \mathbf{b} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}.$$

Here, the matrix A is not square so we cannot rely of Theorem A.62 and instead row-

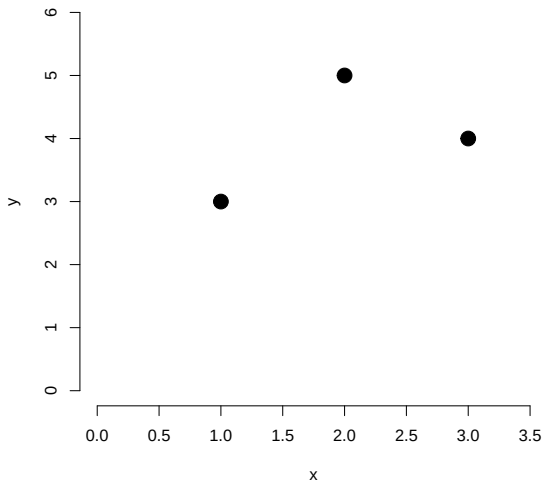


Figure 3.3: A little less ronery!

reduce A. We find

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix} \overset{\substack{R_2 \leftarrow R_2 - R_1 \\ R_3 \leftarrow R_3 - R_1}}{\longrightarrow} \begin{pmatrix} 1 & x_1 \\ 0 & x_2 - x_1 \\ 0 & x_3 - x_1 \end{pmatrix}$$

Thus, we have a solution if all three points lie on the same line, i.e., they are colinear. In all other cases, there is no solution.

So we ask if we can do the next-best thing, which would be to drive a line "as close to as possible" (whatever that means) to the three points, which becomes a problem of **fitting**.

3.2 Least squares problems

We have just established that, given more than two points, the problem we need to consider is that of driving a line as close to as possible to the points. The first problem is to define what we mean by "as close to as possible". From earlier Linear Algebra, you should remember that the shortest distance between a point and a line is the orthogonal projection of the point onto the line. So, given values of a_0 and a_1, we could create the resulting line $y = a_0 + a_1 x$ and compute the distance from the line to each of the three points, sum these distances. Let us do that numerically. To do this, we use projections (Section A.3.7) and more specifically, (A.2).

There is one obvious issue with the previous method: if instead of straight line, we use a more complex curve, then projecting onto the curve becomes more complicated. Indeed, firstly, we need to use the tangent to the curve (and hence need to consider

the derivative of the curve), but also there are no guarantees of uniqueness of the point being the closest to the curve.

Hence, instead of the projection onto the curve, we use another approach that considers the distance between the data points (the known points) and what value we "predict" that point should have when we use the curve that we find.

For future use, let me define a function that returns the value of (3.1):

```
> my_line = function(x, a_0, a_1) {
+     return(a_0 + a_1*x)
+ }
```

In Figure 3.2, you can see the situation when $a_0 = 3.5$ and $a_1 = 0.4$, with the distance being the length of the blue line segments between the red and black dots.

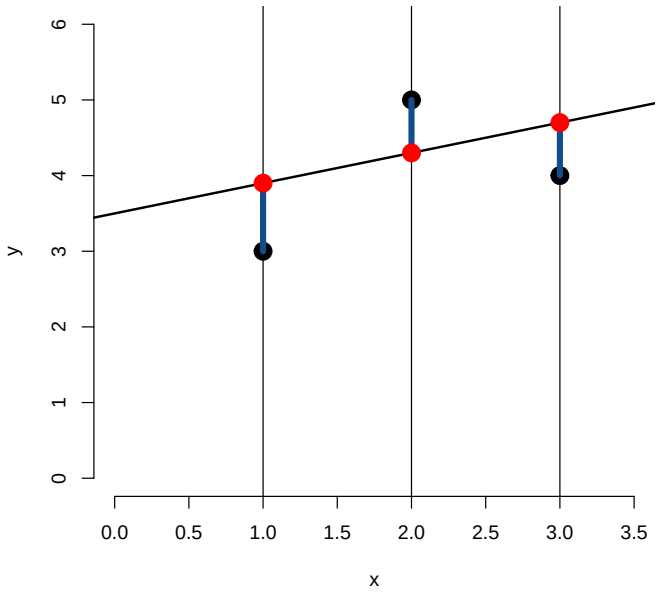


Figure 3.4: Distance between the (given) data points, in black, and the value predicted for the data points if using a line with coefficients $a_0 = 3.5$ and $a_1 = 0.4$.

The location of the black points in Figure 3.2 is easily found: given parameters a_0 and a_1 and the line equation (3.1), they have the same x coordinates as the data points and y coordinates given by, at $x_1 = 1$,

$$\tilde{y}_1 = a_0 + a_1 x_1 = a_0 + a_1,$$

at $x_2 = 2$,

$$\tilde{y}_2 = a_0 + a_1 x_2 = a_0 + 2a_1$$

and at $x_3 = 3$,

$$\tilde{y}_3 = a_0 + a_1 x_3 = a_0 + 3a_1.$$

Consider x_1, for instance. The error we made by using the line with coefficients (a_0, a_1) is $\varepsilon_1 = y_1 - \tilde{y}_1$, where the sign indicates whether we under-estimated or over-estimated the value. We can then form the **error vector**

$$\mathbf{e} = (\varepsilon_1, \varepsilon_2, \varepsilon_3). \tag{3.2}$$

The norm $\|\mathbf{e}\|$ of $\mathbf{e}$ then gives a measure of the overall error we made with our choice of a_0, a_1. Different norms can be used, which will pick up on different characteristics, but in linear least squares, we use the Euclidean norm

$$\|\mathbf{e}\| = \sqrt{\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2},$$

because this norm has very useful properties. Our objective is then to find the coefficients a_0, a_1 such that $\|\mathbf{e}\|$ is minimal. This problem generalises of course to an arbitrary number of data points, giving rise to the following definition.

Definition 3.1 (Linear least squares problem)**.** *Given a collection $(x_1, y_1), \ldots, (x_n, y_n)$ of data points, find the coefficients a_0, a_1 of the line $y = a_0 + a_1 x$ such that the error*

$$\|\mathbf{e}\| = \sqrt{\varepsilon_1^2 + \cdots + \varepsilon_n^2} = \sqrt{(y_1 - \tilde{y}_1)^2 + \cdots + (y_n - \tilde{y}_n)^2} \tag{3.3}$$

is minimal, where $\tilde{y}_i = a_0 + a_1 x_i$ for $i = 1, \ldots, n$.

Before continuing, let us write a function to compute the error when the norm is the Euclidean norm. This function expects that `points` are a list, as we have been using this far.

```
> error = function(a_0, a_1, points) {
+     y_tilde = my_line(points$x, a_0 = a_0, a_1 = a_1)
+     e = points$y - y_tilde
+     return(sqrt(sum(e^2)))
+ }
```

Using it on a couple of different parameter values gives the following.

```
> error(a_0 = 3.5, a_1 = 0.4, points)

[1] 1.337909

> error(a_0 = 3, a_1 = 0.5, points)

[1] 1.224745
```

Of course, it would be Sisyphean to undertake to find a solution by hand: we do not know that a solution to the problem exists and even if it does, a trial and error approach could prove quite lengthy.

3.3 Solving by brute force using a genetic algorithm

We want to minimise the error (3.3). Let us set aside for now the issue of existence of a minimal value for the expression and try to find a numerical solution to this **minimisation problem**, which falls within the larger category of **optimisation problems**. There are many algorithms available to perform optimisation, with probably the most well known being the (deterministic) gradient descent algorithm. Here, we will use a *stochastic* optimisation method called a **genetic algorithm**. The idea is to use a mechanism mimicking evolution's drive towards higher *fitness*. As such, genetic algorithms study the maximisation of functions, but we will see that it is very easy to use them as well with the goal of minimising functions.

The "philosophy" of the genetic algorithm is as follows. Suppose we have a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ of several variables.

1. A point $\mathbf{x} \in \mathbb{R}^n$ is a *gene*.
2. The function value $f(\mathbf{x})$ is the fitness of the gene $\mathbf{x} \in \mathbb{R}^n$.
3. Initiate the algorithm with N (the population size) random genes $\mathbf{x}_1, \ldots, \mathbf{x}_N$.
4. Repeat the following steps for the specified number of generations.
   (a) Evaluate the fitness $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)$ of all the genes in the population.
   (b) Keep a certain number of the best performing genes, i.e., the ones with higher fitness.
   (c) To prepare the population for the next generation $i + 1$, keep these best performing genes and throw in some new genes. Some of these new genes may be random as initially, but most are obtained by using "genetic operations" on the best performing genes in the current generation: crossover, mutations, etc.

This algorithm is known to have good convergence properties. It is also less prone to being stuck at local maxima, because of the stochasticity of the process. Also, this algorithm is highly parallelisable: in a given generation, fitness evaluations are completely independent, i.e., they do not require to know the results of the other evaluations. This means that if you can devote a number of computing cores to the task, they can run independently, greatly speeding up the process.

In `R`, there are several libraries implementing genetic algorithms; `GA` is probably a good place to start. Additionally, `ga`, the main function in that library, makes it almost trivial to parallelise your code, as I will demonstrate later. Assuming the library is installed and loaded, running the algorithm is really simple for the problem we are considering.

```
> GA = ga(type = "real-valued",
+         fitness = function(x) -error(a_0 = x[1], a_1 = x[2], points),
+         lower = c(0, -1), upper = c(10, 1),
+         suggestions = c(a_0 = 3.5, a_1 = 0.4),
+         popSize = 200, maxiter = 150,
+         monitor = FALSE)
```

Let me detail the function call above, as the components can be a little confusing. As explained, genetic algorithms are *maximisers*, so in order to minimise the error, for `fitness` (the function that computes the fitness), we use `function(x) -error(a_0 = x[1], a_1 = x[2], points)`. Note that if `error()` had only depended on `x` and not also on `points`, we could have used `fitness = -error(a_0 = x[1], a_1 = x[2])`; the reason for using `function(x)` is that `error()` is a function of several variables, only a subset of which are of interest to `ga` for its computations.

`lower` and `upper` are lower and upper bounds for the (random) values of parameters. They are vectors that must have the same length; they also determine the size of the genes, i.e., the number of parameters with respect to which we are optimising. `lower`, `upper` and `fitness` are the only required arguments to the function.

Given `lower` and `upper`, those gene components that are chosen at random are chosen at random uniformly between the lower and upper bound. For instance, the call above says that we seek $a_0 \in [0, 10]$ and $a_1 \in [-1, 1]$. There are instances when we want to hasard guesses as to the parameter values; these guesses can be given to the function as a matrix (or a vector if there is only one). Above, I have added the values I used for the figures above, as they did not look too bad.

Finally, `popSize` and `maxiter` have default values of 50 and 100, respectively, which I have increased here just to point out their role. `popSize` is the size of the population, i.e., the number of genes that are evaluated at each generation, while `maxiter` is the maximum number of generations for which the algorithm is run. Increasing both values from their defaults might help with some problems, but they might also vastly increase the run time. And `monitor` is set to `FALSE` to suppress any output from the function, otherwise progress is shown for each generation.

Using the command `plot(GA)`, where `GA` is the result obtained above, gives a visual representation of the behaviour of the algorithm, showing how the mean fitness, minimum and maximum fitness vary through generations. Give it a try (this is not shown here). The result of using `ga` is a so-called S4 object, so accessing the result is done using @ rather than $. The most useful part of the return value is of course the parameters we are looking for, which are returned as follows.

```
> GA@solution

           x1        x2
[1,] 3.000126 0.4999276
```

You could also see the value of the fitness at that solution by looking at `GA@fitnessValue`.

3.4 How about a little finesse?

We just saw how to minimise $\|e\|$ by brute force using a genetic algorithm. Let us now see what mathematics can tell us about this.

For a data point $i = 1, \dots, n$, because of the form of (3.1), the error can be written as

$$\varepsilon_i = y_i - \tilde{y}_i = y_i - (a_0 + a_1 x_i).$$

Do this for all data points,

$$\begin{aligned} \varepsilon_1 &= y_1 - (a_0 + a_1 x_1) \\ &\vdots \\ \varepsilon_n &= y_n - (a_0 + a_1 x_n) \end{aligned}$$

and rewrite in matrix form. This allows to reformulate the problem as follows.

Definition 3.2 (Least squares for an affine function)**.** *Given a set of data points*

$$(x_1, y_1), \dots, (x_n, y_n), \tag{3.4}$$

define the vectors

$$\mathbf{b} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \tag{3.5}$$

and the matrix

$$A = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}. \tag{3.6}$$

Let the error vector $\mathbf{e} = (\varepsilon_1, \dots, \varepsilon_n)^T$ *be defined as*

$$\mathbf{e} = \mathbf{b} - A\mathbf{x}. \tag{3.7}$$

Find $\mathbf{x}$ *such that* $\|\mathbf{e}\|$ *is minimum.*

Note that this is where the complication comes from. Finding $\mathbf{e} = \mathbf{b} - A\mathbf{x}$ is trivial: since $\mathbf{b}$ and A are given, plugging in any value of $\mathbf{x}$ gives a resulting value of $\mathbf{e}$. However, what we seek here is $\mathbf{x}$ that minimises $\|\mathbf{e}\|$, i.e., of all possible values of $\mathbf{x}$, i.e., here, $\mathbb{R}^2$, we want the one or the ones that make $\|\mathbf{e}\|$ the smallest possible. To solve the problem, we need a few more tools.

Definition 3.3 (Least squares solution)**.** *Let* $A \in \mathcal{M}_{mn}$ *and* $\mathbf{b} \in \mathbb{R}^m$*. A* ***least squares solution*** *of* $A\mathbf{x} = \mathbf{b}$ *is a vector* $\tilde{\mathbf{x}} \in \mathbb{R}^n$ *such that*

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad \|\mathbf{b} - A\tilde{\mathbf{x}}\| \leq \|\mathbf{b} - A\mathbf{x}\|. \tag{3.8}$$

In other words, if we denote $\tilde{\mathbf{e}}$ the error corresponding to a least squares solution, $\tilde{\mathbf{e}}$ is such that $\tilde{\mathbf{e}} \leq \mathbf{e}$ for all $\mathbf{x}$.

Definition 3.4 (Best approximation)**.** *Let V be a vector space, $W \subseteq V$ and $\mathbf{v} \in V$. The **best approximation** to $\mathbf{v}$ in W is $\tilde{\mathbf{v}} \in W$ such that*

$$\forall \mathbf{w} \in W, \mathbf{w} \neq \tilde{\mathbf{v}}, \quad \|\mathbf{v} - \tilde{\mathbf{v}}\| < \|\mathbf{v} - \mathbf{w}\|. \tag{3.9}$$

Theorem 3.5 (Best approximation theorem)**.** *Let V be a vector space with an inner product, $W \subset V$ and $\mathbf{v} \in V$. Then $\mathsf{proj}_W(\mathbf{v})$ is the best approximation to $\mathbf{v}$ in W*

Let us reformulate this result even more explicitly in terms of projections.

Theorem 3.6 (Best approximation theorem)**.** *Let V be a vector space with an inner product, $W \subset V$ and $\mathbf{v} \in V$. Then $\mathsf{proj}_W(\mathbf{v}) \in W$ is the best approximation to $\mathbf{v}$ in W, i.e.,*

$$\forall \mathbf{w} \in W, \mathbf{w} \neq \mathsf{proj}_W(\mathbf{v}), \quad \|\mathbf{v} - \mathsf{proj}_W(\mathbf{v})\| < \|\mathbf{v} - \mathbf{w}\|$$

We then have the following theorem.

Theorem 3.7 (Least squares theorem)**.** *Let $A \in \mathcal{M}_{mn}$ and $\mathbf{b} \in \mathbb{R}^m$. Then*

1. *$A\mathbf{x} = \mathbf{b}$ always has at least one least squares solution $\tilde{\mathbf{x}}$*
2. *$\tilde{\mathbf{x}}$ least squares solution to $A\mathbf{x} = \mathbf{b} \iff \tilde{\mathbf{x}}$ is a solution to the **normal equations** $A^T A\tilde{\mathbf{x}} = A^T\mathbf{b}$*
3. *A has linearly independent columns $\iff A^TA$ invertible. In this case, the least squares solution is unique and*

$$\tilde{\mathbf{x}} = \left(A^T A\right)^{-1} A^T\mathbf{b}.$$

Before we prove this, let us consider what Theorem 3.7 says. The first point stipulates that we are always be able to find a least squares solution. The second provides us with a way to construct least squares solutions that always works. The third condition then says that under stricter conditions, the least squares is unique.

Proof. Let us find the least squares solution $\forall \mathbf{x} \in \mathbb{R}^n$, $A\mathbf{x}$ is a vector in the **column space** of A (the space spanned by the vectors making up the columns of A)

Since $\mathbf{x} \in \mathbb{R}^n$, $A\mathbf{x} \in \mathsf{col}(A)$. This implies that the least squares solution of $A\mathbf{x} = \mathbf{b}$ is a vector $\tilde{\mathbf{y}} \in \mathsf{col}(A)$ such that

$$\forall \mathbf{y} \in \mathsf{col}(A), \quad \|\mathbf{b} - \tilde{\mathbf{y}}\| \leq \|\mathbf{b} - \mathbf{y}\|.$$

This looks very much like Best approximation and Best approximation theorem

Putting things together We just stated: The least squares solution of $A\mathbf{x} = \mathbf{b}$ is a vector $\tilde{\mathbf{y}} \in \mathsf{col}(A)$ s.t.

$$\forall \mathbf{y} \in \mathsf{col}(A), \quad \|\mathbf{b} - \tilde{\mathbf{y}}\| \leq \|\mathbf{b} - \mathbf{y}\|$$

$\implies W = \mathsf{col}(A)$, $\mathbf{v} = \mathbf{b}$ and $\tilde{\mathbf{y}} = \mathsf{proj}_{\mathsf{col}(A)}(\mathbf{b})$

So if $\tilde{\mathbf{x}}$ is a least squares solution of $A\mathbf{x} = \mathbf{b}$, then

$$\tilde{\mathbf{y}} = A\tilde{\mathbf{x}} = \mathsf{proj}_{\mathsf{col}(A)}(\mathbf{b})$$

We have

$$\mathbf{b} - A\tilde{\mathbf{x}} = \mathbf{b} - \mathsf{proj}_{\mathsf{col}(A)}(\mathbf{b}) = \mathsf{perp}_{\mathsf{col}(A)}(\mathbf{b})$$

and it is easy to show that

$$\mathsf{perp}_{\mathsf{col}(A)}(\mathbf{b}) \perp \mathsf{col}(A)$$

So for all columns $\mathbf{a}_i$ of A

$$\mathbf{a}_i \cdot (\mathbf{b} - A\tilde{\mathbf{x}}) = 0$$

which we can also write as $\mathbf{a}_i^T(\mathbf{b} - A\tilde{\mathbf{x}}) = 0$

For all columns $\mathbf{a}_i$ of A,

$$\mathbf{a}_i^T(\mathbf{b} - A\tilde{\mathbf{x}}) = 0$$

This is equivalent to saying that

$$A^T(\mathbf{b} - A\tilde{\mathbf{x}}) = \mathbf{0}$$

We have

$$\begin{aligned} A^T(\mathbf{b} - A\tilde{\mathbf{x}}) = \mathbf{0} &\iff A^T\mathbf{b} - A^TA\tilde{\mathbf{x}} = \mathbf{0} \\ &\iff A^T\mathbf{b} = A^TA\tilde{\mathbf{x}} \\ &\iff A^TA\tilde{\mathbf{x}} = A^T\mathbf{b} \end{aligned}$$

The latter system constitutes the **normal equations** for $\tilde{\mathbf{x}}$

We have seen 1 and 2, we will not show 3 (it is not hard) □

3.5 Fitting something more complicated

Suppose we want to fit something a bit more complicated, because the data does not seem to be really distributed along a straight line. For instance, instead of the affine function

$$y = a_0 + a_1x,$$

suppose we want to fit the quadratic

$$y = a_0 + a_1x + a_2x^2$$

or even the exponential

$$y = k_0e^{k_1x}.$$

How should we proceed?

3.5.1 Fitting the quadratic

We have the data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and want to fit

$$y = a_0 + a_1 x + a_2 x^2.$$

At (x_1, y_1),

$$\tilde{y}_1 = a_0 + a_1 x_1 + a_2 x_1^2.$$

$\vdots$

At (x_n, y_n),

$$\tilde{y}_n = a_0 + a_1 x_n + a_2 x_n^2.$$

In terms of the error

$$\begin{aligned} \varepsilon_1 &= y_1 - \tilde{y}_1 = y_1 - (a_0 + a_1 x_1 + a_2 x_1^2) \\ &\vdots \\ \varepsilon_n &= y_n - \tilde{y}_n = y_n - (a_0 + a_1 x_n + a_2 x_n^2) \end{aligned}$$

i.e.,

$$\mathbf{e} = \mathbf{b} - A\mathbf{x}$$

where

$$\mathbf{e} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}, A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \text{ and } \mathbf{b} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Theorem 3.7 applies, with here $A \in \mathcal{M}_{n3}$ and $\mathbf{b} \in \mathbb{R}^n$

Recall that the points are (1,3), (2,5) and (3,4). Let us fit $y = a_0 + a_1 x + a_2 x^2$. The matrix A takes the form

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}$$

b takes the form

$$b = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

3.5.2 Fitting the exponential

Things are a bit more complicated when trying to fit the data to an exponential function $y = k_0 \exp(k_1 x)$. Indeed, proceeding as we did before, we get the system

$$\begin{aligned} y_1 &= k_0 e^{k_1 x_1} \\ &\vdots \\ y_n &= k_0 e^{k_1 x_n}. \end{aligned}$$

The $e^{k_1 x_i}$ are nonlinear terms, they cannot be put in a matrix. However, this can be transformed into a form more amenable to the technique we already used. Indeed, take the ln of both sides of the equation $y_i = k_0 \exp(k_1 x_i)$:

$$\begin{aligned} \ln(y_i) &= \ln(k_0 e^{k_1 x_i}) \\ &= \ln(k_0) + \ln(e^{k_1 x_i}) \\ &= \ln(k_0) + k_1 x_i. \end{aligned}$$

If $y_i, k_0 > 0$, then their ln are defined and we are back essentially to the affine case, with the system being

$$\mathbf{y} = A\mathbf{x} + \mathbf{b}$$

where

$$A = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \mathbf{x} = \begin{pmatrix} k_1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \ln(k_0) \end{pmatrix} \text{ and } \mathbf{y} = \begin{pmatrix} \ln(y_1) \\ \vdots \\ \ln(y_n) \end{pmatrix}.$$

Chapter 4

Matrix factorisations

Matrix factorisations, which are known under many different other names, are theoretical results and algorithms that allow to write matrices typically as the product of several matrices with specified forms. The matrices in the factorisation usually have known properties that make them much easier to perform computations with.

There are several different types of factorisations. Here, we study two: the QR factorisation and the SVD. Both have many applications in their own right, but we also return later to the problem of least squares. Then, we present another dimensionality reduction technique, principal component analysis (PCA).

Before we start, though, we need to learn a little about *orthogonal sets* and *orthogonal matrices*.

4.1 Orthogonal matrices

Definition 4.1 (Orthogonal set of vectors)**.** *The set of vectors* $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \in \mathbb{R}^n$ *is an* ***orthogonal set*** *if*

$$\forall i, j = 1, \dots, k, \quad i \neq j \implies \mathbf{v}_i \bullet \mathbf{v}_j = 0$$

Theorem 4.2. $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \in \mathbb{R}^n$ *with* $\forall i$, $\mathbf{v}_i \neq \mathbf{0}$*, orthogonal set* $\implies$ $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \in \mathbb{R}^n$ *linearly independent.*

Proof. Assume $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ orthogonal set with $\mathbf{v}_i \neq \mathbf{0}$ for all $i = 1, \dots, k$. Recall $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is LI if

$$c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k = \mathbf{0} \iff c_1 = \cdots = c_k = 0$$

So assume $c_1, \dots, c_k \in \mathbb{R}$ are s.t. $c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k = \mathbf{0}$. Recall that $\forall \mathbf{x} \in \mathbb{R}^k$, $\mathbf{0}_k \bullet \mathbf{x} = 0$. So for some $\mathbf{v}_i \in \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$

$$\begin{aligned} 0 &= \mathbf{0} \bullet \mathbf{v}_i \\ &= (c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k) \bullet \mathbf{v}_i \\ &= c_1\mathbf{v}_1 \bullet \mathbf{v}_i + \cdots + c_k\mathbf{v}_k \bullet \mathbf{v}_i \end{aligned} \tag{4.1}$$

As $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ orthogonal, $\mathbf{v}_j \bullet \mathbf{v}_i = 0$ when $i \neq j$, (4.1) reduces to

$$c_i \mathbf{v}_i \bullet \mathbf{v}_i = 0 \iff c_i \|\mathbf{v}_i\|^2 = 0$$

As $\mathbf{v}_i \neq 0$ for all i, $\|\mathbf{v}_i\| \neq 0$ and so $c_i = 0$. This is true for all i, hence the result. □

Definition 4.3 (Orthogonal basis)**.** *Let S be a basis of the subspace $W \subset \mathbb{R}^n$ composed of an orthogonal set of vectors. We say S is an* ***orthogonal basis*** *of W*

Example – Vectors of the standard basis of $\mathbb{R}^3$ For $\mathbb{R}^3$, we denote

$$\mathbf{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \mathbf{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

For $\mathbb{R}^k$, $k > 3$, we denote them $\mathbf{e}_i$. Clearly, $\{\mathbf{i}, \mathbf{j}\}$, $\{\mathbf{i}, \mathbf{k}\}$, $\{\mathbf{j}, \mathbf{k}\}$ and $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ are orthogonal sets. The standard basis vectors are also $\neq \mathbf{0}$, so the sets are linearly independent. And

$$\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$$

is an orthogonal basis of $\mathbb{R}^3$ since it spans $\mathbb{R}^3$ and is linearly independent. Then any point (or vector) $(c_1, c_2, c_3) \in \mathbb{R}^3$ can be written as

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = c_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = c_1\mathbf{i} + c_2\mathbf{j} + c_3\mathbf{k},$$

with this linear combination being unique.

There is an orthonormal version of these definitions and results.

Definition 4.4 (Orthonormal set)**.** *The set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \in \mathbb{R}^n$ is an* ***orthonormal set*** *if it is an orthogonal set and furthermore*

$$\forall i = 1, \ldots, k, \quad \|\mathbf{v}_i\| = 1.$$

Definition 4.5 (Orthonormal basis)**.** *A basis of the subspace $W \subset \mathbb{R}^n$ is an* ***orthonormal basis*** *if the vectors composing it are an orthonormal set.*

$\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \in \mathbb{R}^n$ is orthonormal if

$$\mathbf{v}_i \bullet \mathbf{v}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.6. *Let $Q \in \mathcal{M}_{mn}$. The columns of Q form an orthonormal set if and only if*

$$Q^T Q = \mathbb{I}_n$$

Definition 4.7 (Orthogonal matrix)**.** *$Q \in \mathcal{M}_n$ is an **orthogonal matrix** if its columns form an orthonormal set*

So $Q \in \mathcal{M}_n$ orthogonal if $Q^TQ = \mathbb{I}$, i.e., $Q^T = Q^{-1}$

Theorem 4.8 (NSC for orthogonality)**.** *$Q \in \mathcal{M}_n$ orthogonal $\iff$ $Q^{-1} = Q^T$*

Theorem 4.9 (Orthogonal matrices "encode" isometries)**.** *Let $Q \in \mathcal{M}_n$. The following are equivalent.*

1. *Q orthogonal.*
2. *$\forall \mathbf{x} \in \mathbb{R}^n$, $\|Q\mathbf{x}\| = \|\mathbf{x}\|$.*
3. *$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $Q\mathbf{x} \bullet Q\mathbf{y} = \mathbf{x} \bullet \mathbf{y}$.*

Theorem 4.10. *Let $Q \in \mathcal{M}_n$ be orthogonal. Then*

1. *The rows of Q form an orthonormal set.*
2. *Q^{-1} orthogonal.*
3. *$\det Q = \pm 1$.*
4. *$\forall \lambda \in \sigma(Q)$, $|\lambda| = 1$.*
5. *If $Q_2 \in \mathcal{M}_n$ also orthogonal, then QQ_2 orthogonal.*

Proof. (Proof of 4 in Theorem 4.10) All statements in Theorem 4.10 are easy, but let's focus on 4

Let λ be an eigenvalue of $Q \in \mathcal{M}_n$ orthogonal, i.e., $\exists \mathbb{R}^n \ni \mathbf{x} \neq \mathbf{0}$ s.t.

$$Q\mathbf{x} = \lambda\mathbf{x}$$

Take the norm on both sides

$$\|Q\mathbf{x}\| = \|\lambda\mathbf{x}\|$$

From 2 in Theorem 4.9, $\|Q\mathbf{x}\| = \|\mathbf{x}\|$ and from the properties of norms, $\|\lambda\mathbf{x}\| = |\lambda| \, \|\mathbf{x}\|$, so we have

$$\|Q\mathbf{x}\| = \|\lambda\mathbf{x}\| \iff \|\mathbf{x}\| = |\lambda| \, \|\mathbf{x}\| \iff 1 = |\lambda|$$

(we can divide by $\|\mathbf{x}\|$ since $\mathbf{x} \neq \mathbf{0}$ as an eigenvector) □

4.2 The Gram-Schmidt orthonormalisation procedure

We could spend a long time on these interesting notions, but we need to get back to the QR decomposition. What this aims to do is to construct an orthogonal basis for a subspace $W \subset \mathbb{R}^n$. To do this, we use the *Gram-Schmidt orthogonalisation process*, which turns a basis of W into an orthogonal basis of W.

4.2.1 Projections onto subspaces

Definition 4.11 (Orthogonal projection onto a subspace)**.** *$W \subset \mathbb{R}^n$ a subspace and $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ an orthogonal basis of W. $\forall \mathbf{v} \in \mathbb{R}^n$, the **orthogonal projection** of $\mathbf{v}$ **onto** W is*

$$\mathsf{proj}_W(\mathbf{v}) = \frac{\mathbf{u}_1 \bullet \mathbf{v}}{\|\mathbf{u}_1\|^2}\mathbf{u}_1 + \cdots + \frac{\mathbf{u}_k \bullet \mathbf{v}}{\|\mathbf{u}_k\|^2}\mathbf{u}_k$$

Definition 4.12 (Component orthogonal to a subspace)**.** *$W \subset \mathbb{R}^n$ a subspace and $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ an orthogonal basis of W. $\forall \mathbf{v} \in \mathbb{R}^n$, the **component** of $\mathbf{v}$ **orthogonal to** W is*

$$\mathsf{perp}_W(\mathbf{v}) = \mathbf{v} - \mathsf{proj}_W(\mathbf{v})$$

4.2.2 The Gram-Schmidt process

Theorem 4.13. *$W \subset \mathbb{R}^n$ a subset and $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ a basis of W. Let*

$$\begin{aligned}
\mathbf{v}_1 &= \mathbf{x}_1 \\
\mathbf{v}_2 &= \mathbf{x}_2 - \frac{\mathbf{v}_1 \bullet \mathbf{x}_2}{\|\mathbf{v}_1\|^2}\mathbf{v}_1 \\
\mathbf{v}_3 &= \mathbf{x}_3 - \frac{\mathbf{v}_1 \bullet \mathbf{x}_3}{\|\mathbf{v}_1\|^2}\mathbf{v}_1 - \frac{\mathbf{v}_2 \bullet \mathbf{x}_3}{\|\mathbf{v}_2\|^2}\mathbf{v}_2 \\
&\vdots \\
\mathbf{v}_k &= \mathbf{x}_k - \frac{\mathbf{v}_1 \bullet \mathbf{x}_k}{\|\mathbf{v}_1\|^2}\mathbf{v}_1 - \cdots - \frac{\mathbf{v}_{k-1} \bullet \mathbf{x}_k}{\|\mathbf{v}_{k-1}\|^2}\mathbf{v}_{k-1}
\end{aligned}$$

and

$$W_1 = \mathsf{span}(\mathbf{x}_1), W_2 = \mathsf{span}(\mathbf{x}_1, \mathbf{x}_2), \ldots, W_k = \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$$

Then $\forall i = 1, \ldots, k$, $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$ orthogonal basis for W_i

4.3 The QR factorisation

Theorem 4.14. *Let $A \in \mathcal{M}_{mn}$, with A having linearly independent. Then A can be factored as*

$$A = QR, \tag{4.2}$$

where $Q \in \mathcal{M}_{mn}$ has orthonormal columns and $R \in \mathcal{M}_n$ is a nonsingular upper triangular matrix.

4.3.1 Back to least squares

So what was the point of all that..?

Theorem 4.15 (Least squares with QR factorisation)**.** *$A \in \mathcal{M}_{mn}$ with LI columns, $\mathbf{b} \in \mathbb{R}^m$. If $A = QR$ is a QR factorisation of A, then the unique least squares solution $\tilde{\mathbf{x}}$ of $A\mathbf{x} = \mathbf{b}$ is*

$$\tilde{\mathbf{x}} = R^{-1}Q^T\mathbf{b}$$

Proof of Theorem 4.15 A has LI columns so

- least squares $A\mathbf{x} = \mathbf{b}$ has unique solution $\tilde{\mathbf{x}} = (A^TA)^{-1}A^T\mathbf{b}$
- by Theorem 4.14, A can be written as $A = QR$ with $Q \in \mathcal{M}_{mn}$ with orthonormal columns and $R \in \mathcal{M}_n$ nonsingular and upper triangular

So

$$\begin{aligned} A^TA\tilde{\mathbf{x}} = A^T\mathbf{b} &\implies (QR)^TQR\tilde{\mathbf{x}} = (QR)^T\mathbf{b} \\ &\implies R^TQ^TQR\tilde{\mathbf{x}} = R^TQ^T\mathbf{b} \\ &\implies R^T\mathbb{I}_nR\tilde{\mathbf{x}} = R^TQ^T\mathbf{b} \\ &\implies R^TR\tilde{\mathbf{x}} = R^TQ^T\mathbf{b} \\ &\implies (R^T)^{-1}R\tilde{\mathbf{x}} = (R^T)^{-1}R^TQ^T\mathbf{b} \\ &\implies R\tilde{\mathbf{x}} = Q^T\mathbf{b} \\ &\implies \tilde{\mathbf{x}} = R^{-1}Q^T\mathbf{b} \end{aligned}$$

4.4 The singular values decomposition (SVD)

The singular value decomposition, known mostly by its acronym SVD, is another type of factorisation.

Definition 4.16 (Singular value)**.** *Let $A \in \mathcal{M}_{mn}(\mathbb{R})$. The **singular values** of A are the real numbers*

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$$

that are the square roots of the eigenvalues of A^TA.

Recall that $\forall A \in \mathcal{M}_{mn}$, the matrix A^TA is symmetric (see Theorem A.31). Since A is real, there furthermore holds that A^TA has all its eigenvalues real and nonnegative (by Theorem A.33). (Note that the proofs of these results are important, so refer to Section A.6.1 for details.) As a consequence, this definition is valid (it would not were some of the eigenvalues complex).

Theorem 4.17 (Singular value decomposition)**.** *$A \in \mathcal{M}_{mn}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$. Then there exists $U \in \mathcal{M}_m$ orthogonal, $V \in \mathcal{M}_n$ orthogonal and a block matrix $\Sigma \in \mathcal{M}_{mn}$ taking the form*

$$\Sigma = \begin{pmatrix} D & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{pmatrix},$$

where

$$D = \mathsf{diag}(\sigma_1, \ldots, \sigma_r) \in \mathcal{M}_r$$

such that

$$A = U\Sigma V^T.$$

Note that by construction, U and V^T are *rotation* or *reflection* matrices, while Σ is a *scaling* matrix.

Definition 4.18. *We call a factorisation as in Theorem 4.17 the* ***singular value decomposition*** *of A. The columns of U and V are, respectively, the* ***left*** *and* ***right singular vectors*** *of A.*

The following result tells us how we can reconstruct the original matrix from its SVD. This is useful in the image compression example of Section 4.5.

Theorem 4.19 (Outer product form of the SVD)**.** *$A \in \mathcal{M}_{mn}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$, $\mathbf{u}_1, \ldots, \mathbf{u}_r$ and $\mathbf{v}_1, \ldots, \mathbf{v}_r$, respectively, left and right singular vectors of A corresponding to these singular values. Then*

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T.$$

4.4.1 Computing the SVD (case of $\neq$ eigenvalues)

To compute the SVD, we use the following result

Theorem 4.20. *Let $A \in \mathcal{M}_n$ symmetric, $(\lambda_1, \mathbf{u}_1)$ and $(\lambda_2, \mathbf{u}_2)$ be eigenpairs, assuming that $\lambda_1 \neq \lambda_2$. Then $\mathbf{u}_1 \bullet \mathbf{u}_2 = 0$*

Proof. Let $A \in \mathcal{M}_n$ be symmetric, $(\lambda_1, \mathbf{u}_1)$ and $(\lambda_2, \mathbf{u}_2)$ be eigenpairs of A with $\lambda_1 \neq \lambda_2$. Then we have

$$\begin{aligned}
\lambda_1(\mathbf{v}_1 \bullet \mathbf{v}_2) &= (\lambda_1 \mathbf{v}_1) \bullet \mathbf{v}_2 \\
&= A\mathbf{v}_1 \bullet \mathbf{v}_2 \\
&= (A\mathbf{v}_1)^T \mathbf{v}_2 \\
&= \mathbf{v}_1^T A^T \mathbf{v}_2 \\
&= \mathbf{v}_1^T (A\mathbf{v}_2) [A \text{ symmetric so } A^T = A] \\
&= \mathbf{v}_1^T (\lambda_2 \mathbf{v}_2) \\
&= \lambda_2 (\mathbf{v}_1^T \mathbf{v}_2) \\
&= \lambda_2 (\mathbf{v}_1 \bullet \mathbf{v}_2).
\end{aligned}$$

It follows that $(\lambda_1 - \lambda_2)(\mathbf{v}_1 \bullet \mathbf{v}_2) = 0$. However, since $\lambda_1 \neq \lambda_2$, this means that $\mathbf{v}_1 \bullet \mathbf{v}_2 = 0$. □

4.4.2 Computing the SVD (case of $\neq$ eigenvalues)

If all eigenvalues of A^TA are distinct, we can use Theorem 4.20

1. Compute $A^TA \in \mathcal{M}_n$
2. Compute eigenvalues $\lambda_1, \ldots, \lambda_n$ of A^TA; order them as $\lambda_1 > \cdots > \lambda_n \geq 0$ ($>$ not $\geq$ since $\neq$)
3. Compute singular values $\sigma_1 = \sqrt{\lambda_1}, \ldots, \sigma_n = \sqrt{\lambda_n}$
4. Diagonal matrix D in Σ is either in $\mathcal{M}_n$ (if $\sigma_n > 0$) or in $\mathcal{M}_{n-1}$ (if $\sigma_n = 0$)
5. Since eigenvalues are distinct, Theorem 4.20 $\implies$ eigenvectors are orthogonal set. Compute these eigenvectors in the same order as the eigenvalues
6. Normalise them and use them to make the matrix V, i.e., $V = [\mathbf{v}_1 \cdots \mathbf{v}_n]$
7. To find the $\mathbf{u}_i$, compute, for $i = 1, \ldots, r$,

$$\mathbf{u}_i = \frac{1}{\sigma_i} A\mathbf{v}_i$$

   and ensure that $\|\mathbf{u}_i\| = 1$

4.4.3 Computing the SVD (case where some eigenvalues are =)

1. Compute $A^TA \in \mathcal{M}_n$
2. Compute eigenvalues $\lambda_1, \ldots, \lambda_n$ of A^TA; order them as $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$
3. Compute singular values $\sigma_1 = \sqrt{\lambda_1}, \ldots, \sigma_n = \sqrt{\lambda_n}$, with $r \leq n$ the index of the last positive singular value
4. For eigenvalues that are distinct, proceed as before
5. For eigenvalues with multiplicity > 1, we need to ensure that the resulting eigenvectors are LI *and* orthogonal

Dealing with eigenvalues with multiplicity > 1 When an eigenvalue has (algebraic) multiplicity > 1, e.g., characteristic polynomial contains a factor like $(\lambda - 2)^2$, things can become a little bit more complicated

The proper way to deal with this involves the so-called Jordan Normal Form (another matrix decomposition)

In short: not all square matrices are diagonalisable, but all square matrices admit a JNF

Sometimes, we can find several LI eigenvectors associated to the same eigenvalue. Check this. If not, need to use the following

Definition 4.21 (Generalised eigenvectors)**.** $\mathbf{x} \neq \mathbf{0}$ ***generalized eigenvector*** *of rank m of $A \in \mathcal{M}_n$ corresponding to eigenvalue λ if*

$$(A - \lambda\mathbb{I})^m \mathbf{x} = \mathbf{0}$$

but

$$(A - \lambda\mathbb{I})^{m-1} \mathbf{x} \neq \mathbf{0}$$

Procedure for generalised eigenvectors $A \in \mathcal{M}_n$ and assume λ eigenvalue with algebraic multiplicity k

Find $\mathbf{v}_1$, "classic" eigenvector, i.e., $\mathbf{v}_1 \neq \mathbf{0}$ s.t. $(A - \lambda\mathbb{I})\mathbf{v}_1 = \mathbf{0}$

Find generalised eigenvector $\mathbf{v}_2$ of rank 2 by solving for $\mathbf{v}_2 \neq \mathbf{0}$,

$$(A - \lambda\mathbb{I})\mathbf{v}_2 = \mathbf{v}_1$$

...

Find generalised eigenvector $\mathbf{v}_k$ of rank k by solving for $\mathbf{v}_k \neq \mathbf{0}$,

$$(A - \lambda\mathbb{I})\mathbf{v}_k = \mathbf{v}_{k-1}$$

Then $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ LI

Back to the normal procedure With the LI eigenvectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ corresponding to λ

Apply Gram-Schmidt to get orthogonal set

For all eigenvalues with multiplicity > 1, check that you either have LI eigenvectors or do what we just did

When you are done, be back on your merry way to step 6 in the case where eigenvalues are all $\neq$

I am caricaturing a little here: there can be cases that do not work exactly like this, but this is general enough..

4.5 Compressing images

This example is adapted from ... To summarise, we consider an image that, for simplicity, we assume is in shades of grey. Such an image can be stored in a matrix $A \in \mathcal{M}_{mn}$ in which, typically, a value of 0 means the pixel is black and a value of 1 means it is white, with grey scale covering all the range in between. Take the SVD of A. Then the small singular values carry information about the regions with little variation and can perhaps be omitted, whereas the large singular values carry information about more "dynamic" regions of the image. To see this, we use the outer product form of the SVD given by Theorem 4.19 to reconstruct an image from its SVD.

Namely, if we suppose $A \in \mathcal{M}_{mn}$ has r nonzero singular values, then for $k \leq r$, we let

$$A_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T. \tag{4.3}$$

For $k = r$, we have the usual outer product form given in Theorem 4.19 and we fully recuperate A. Using a partial sum ($k < r$) gives some (but not all) information and the current example illustrates this.

We need two libraries to deal with the image, `bmp` and `pixmap`, which I assume here are installed and loaded. Let us read in the image using `read.bmp` (from the `bmp` library) and transform it to grey scale using `pixmapGrey` (from the `pixmap` library).

```
> orig_image = read.bmp("FIGS/Julien-and-friend.bmp")
> orig_image = pixmapGrey(orig_image)
```

As usual in `R`, the `plot` function is very versatile and adapts to this new type of object, so `plot(my_image)` produces

For context, I "met" this friend in the Natural History Museum of Vienna (the Naturhistorisches Museum Wien). Very nice museum to visit if you are ever in the area. (At the same time, you had better be in the area for a while as there is no shortage of very nice museums to visit in Vienna.) Let us show some information about the image.

```
> orig_image
```

```
Pixmap image
  Type          : pixmapGrey
  Size          : 1000x890
  Resolution    : 1x1
  Bounding box  : 0 0 890 1000
```

The pixelmap (the matrix of greyscale values) is stored in `my_image@grey`, so we set

```
> M = orig_image@grey
```

From now on, we can forget about the image itself and focus on the matrix M.

4.5.1 Doing things "by hand"

4.5.2 Doing things using proper functions

R has, as can be expected from a language originating in statistics, a native function to perform an SVD, which is, very surprisingly, called svd. We use this function and create a function to perform an image compression.

```
> compress_image = function(im, n) {
+   M = svd(im@grey)
+   if (n > length(M$d)) {
+     n = length(im$d)
+   }
+   d_tmp = M$d[1:n]
+   u_tmp = M$u[,1:n]
+   v_tmp = M$v[,1:n]
+   out = list()
+   out$img = mat.or.vec(nr = dim(M$u)[1], nc = dim(M$v)[1])
+   for (i in 1:n) {
+     out$img = out$img + d_tmp[i] * u_tmp[,i] %*% t(v_tmp[,i])
+   }
+   if (min(min(out$img)) < 0 ) {
+     out$img = out$img - min(min(out$img))
+   }
+   out$img = out$img / max(max(out$img))
+   out$nb_pixels_original = dim(im@grey)[1] * dim(im@grey)[2]
+   out$nb_pixels_compressed =
+     length(d_tmp) + dim(u_tmp)[1]*dim(u_tmp)[2] +
+     dim(v_tmp)[1]*dim(v_tmp)[2]
+   out$pct_of_original =
+     out$nb_pixels_compressed / out$nb_pixels_original * 100
+   return(out)
+ }
```

Let me explain what this function does. The arguments to the function are the image to process im and the number n of singular values to use in the compression. The image im is assumed to already be in the form of a greyscale pixelmap. We first idiot-proof the code: if the number of SVD to use is larger than the number possible, we set n to that maximum value.

Let us use the function with 10 singular values. Note that we need to copy the image and then substitute the result of the function to the pixelmap of the copied image. Indeed, the returned matrix cannot simply be displayed as an image.

```
> compressed_image = orig_image
```

```
> result_tmp = compress_image(orig_image, 10)
> compressed_image@grey = result_tmp$img
```

If you did not know the original image, you might find it hard to recognise much of anything, but knowing it, you can probably recognise many features. And this achieved by keeping 10 singular values and their corresponding singular vectors. The function `compress_image` returns this information as `$pct_of_original`.

Chapter 5

Principal component analysis (PCA)

One of the reasons the SVD is used is for dimensionality reduction, as we have seen for instance in the image compression example. Now let us consider another dimensionality reduction technique, the so-called principal components analysis or PCA for short. PCA is often used as a blackbox technique, here we take a look at the mathematics behind it. PCA is another linear algebraic technique, that helps reduce a complex dataset to a lower dimensional one. It is a non-parametric method that does not assume anything about data distribution (from the statistical point of view).

5.1 Brief "review" of some probability concepts

A proper treatment of *probability* requires to use *measure theory* and is not the object of this course. Here, we present just what is needed to understand the content of this chapter. For instance, a **random variable** X is a *measurable* function $X : \Omega \to E$, where Ω is a set of outcomes (*sample space*) and E is a measurable space and we define the probability as follows:

$$\mathbb{P}(X \in S \subseteq E) = \mathbb{P}(\omega \in \Omega | X(\omega) \in S).$$

However, let us not worry about the detailed specifics here.

Distribution function of a r.v., $F(x) = \mathbb{P}(X \leq x)$, describes the distribution of a r.v.

R.v. can be discrete or continuous or .. other things.

Definition 5.1 (Variance)**.** *Let X be a random variable. The **variance** of X is given by*

$$\mathsf{Var}\ X = E\left[(X - E(X))^2\right]$$

where E is the expected value

Definition 5.2 (Covariance)**.** *Let X, Y be jointly distributed random variables. The **covariance** of X and Y is given by*

$$\mathsf{cov}(X, Y) = E\left[(X - E(X))\,(Y - E(Y))\right]$$

Note that $\mathsf{cov}(X, X) = E\left[(X - E(X))^2\right] = \mathsf{Var}\ X$

In practice: "true law" versus "observation" In statistics: we reason on the *true law* of distributions, but we usually have only access to a sample

We then use **estimators** to .. estimate the value of a parameter, e.g., the mean, variance and covariance

Definition 5.3 (Unbiased estimators of the mean and variance)**.** *Let $x_1, \ldots, x_n$ be data points (the* sample*) and*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

*be the **mean** of the data. An unbiased estimator of the variance of the sample is*

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Definition 5.4 (Unbiased estimator of the covariance)**.** *Let $(x_1, y_1), \ldots, (x_n, y_n)$ be data points,*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

be the means of the data. An estimator of the covariance of the sample is

$$\mathsf{cov}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

What does covariance do? Variance explains how data disperses around the mean, in a 1-D case

Covariance measures the relationship between two dimensions. E.g., height and weight

More than the exact value, the sign is important:

- $\mathsf{cov}(X, Y) > 0$: both dimensions change in the same "direction"; e.g., larger height usually means higher weight
- $\mathsf{cov}(X, Y) < 0$: both dimensions change in reverse directions; e.g., time spent on social media and performance in this class
- $\mathsf{cov}(X, Y) = 0$: the dimensions are independent from one another; e.g., sex/gender and "intelligence"

The covariance matrix Typically, we consider more than 2 variables..

Definition 5.5. *Suppose p random variables $X_1, \ldots, X_p$. Then the covariance matrix is the symmetric matrix*

$$\begin{pmatrix} \mathsf{cov}(X_1, X_1) & \mathsf{cov}(X_1, X_2) & \cdots & \mathsf{cov}(X_1, X_p) \\ \mathsf{cov}(X_2, X_1) & \mathsf{cov}(X_2, X_2) & \cdots & \mathsf{cov}(X_2, X_p) \\ \vdots & \vdots & & \vdots \\ \mathsf{cov}(X_p, X_1) & \mathsf{cov}(X_p, X_2) & \cdots & \mathsf{cov}(X_p, X_p) \end{pmatrix}$$

i.e., using the properties of covariance,

$$\begin{pmatrix} \mathsf{Var}\ X_1 & \mathsf{cov}(X_1, X_2) & \cdots & \mathsf{cov}(X_1, X_p) \\ \mathsf{cov}(X_1, X_2) & \mathsf{Var}\ X_2 & \cdots & \mathsf{cov}(X_2, X_p) \\ \vdots & \vdots & & \vdots \\ \mathsf{cov}(X_1, X_p) & \mathsf{cov}(X_2, X_p) & \cdots & \mathsf{Var}\ X_p \end{pmatrix}$$

We want to find what carries the most information

For this, we are going to project the information in a new basis in which the first "dimension" will carry most variance, the second dimension will carry a little less, etc.

In order to do so, we need to learn how to change bases

Change of basis

Definition 5.6 (Change of basis matrix)**.** *$\mathcal{B} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ and $\mathcal{C} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ bases of vector space V*

*The **change of basis matrix** $P_{\mathcal{C}\leftarrow\mathcal{B}} \in \mathcal{M}_n$,*

$$P_{\mathcal{C}\leftarrow\mathcal{B}} = [[\mathbf{u}_1]_{\mathcal{C}} \cdots [\mathbf{u}_n]_{\mathcal{C}}]$$

has columns the coordinate vectors $[\mathbf{u}_1]_{\mathcal{C}}, \ldots, [\mathbf{u}_n]_{\mathcal{C}}$ of the vectors in $\mathcal{B}$ with respect to $\mathcal{C}$

Theorem 5.7. *$\mathcal{B} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ and $\mathcal{C} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ bases of vector space V and $P_{\mathcal{C}\leftarrow\mathcal{B}}$ a change of basis matrix from $\mathcal{B}$ to $\mathcal{C}$*

1. *$\forall \mathbf{x} \in V$, $P_{\mathcal{C}\leftarrow\mathcal{B}}[\mathbf{x}]_{\mathcal{B}} = [\mathbf{x}]_{\mathcal{C}}$*
2. *$P_{\mathcal{C}\leftarrow\mathcal{B}}$ s.t. $\forall \mathbf{x} \in V$, $P_{\mathcal{C}\leftarrow\mathcal{B}}[\mathbf{x}]_{\mathcal{B}} = [\mathbf{x}]_{\mathcal{C}}$ is **unique***
3. *$P_{\mathcal{C}\leftarrow\mathcal{B}}$ invertible and $P^{-1}_{\mathcal{C}\leftarrow\mathcal{B}} = P_{\mathcal{B}\leftarrow\mathcal{C}}$*

Row-reduction method for changing bases

Theorem 5.8. *$\mathcal{B} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ and $\mathcal{C} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ bases of vector space V. Let $\mathcal{E}$ be any basis for V,*

$$B = [[\mathbf{u}_1]_{\mathcal{E}}, \ldots, [\mathbf{u}_n]_{\mathcal{E}}] \text{ and } C = [[\mathbf{v}_1]_{\mathcal{E}}, \ldots, [\mathbf{v}_n]_{\mathcal{E}}]$$

and let $[C|B]$ be the augmented matrix constructed using C and B. Then

$$RREF\left([C|B]\right) = [\mathbb{I}|P_{\mathcal{C}\leftarrow\mathcal{B}}]$$

If working in $\mathbb{R}^n$, this is quite useful with $\mathcal{E}$ the standard basis of $\mathbb{R}^n$ (it does not matter if $\mathcal{B} = \mathcal{E}$)

So the question now becomes

How do we find what new basis to look at our data in?

(Changing the basis does not change the data, just the view you have of it)

(Think of what happens when you do a headstand.. your up becomes down, your right and left switch, but the world does not change, just your view of it)

(Changes of bases are *fundamental* operations in Science)

Setting things up I will use notation (mostly) as in Joliffe's *Principal Component Analysis* (PDF of older version available for free from UofM Libraries)

$\mathbf{x} = (x_1, \ldots, x_p)$ vector of p random variables

We seek a linear function $\mathbf{m}\alpha_1^T\mathbf{x}$ with maximum variance, where $\mathbf{m}\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1p})$, i.e.,

$$\mathbf{m}\alpha_1^T\mathbf{x} = \sum_{j=1}^{p} \alpha_{1j}x_j$$

Then we seek a linear function $\mathbf{m}\alpha_2^T\mathbf{x}$ with maximum variance, uncorrelated to $\mathbf{m}\alpha_1^T\mathbf{x}$

And we continue...

At kth stage, we find a linear function $\mathbf{m}\alpha_k^T\mathbf{x}$ with maximum variance, uncorrelated to $\mathbf{m}\alpha_1^T\mathbf{x}, \ldots, \mathbf{m}\alpha_{k-1}^T\mathbf{x}$

$\mathbf{m}\alpha_i^T\mathbf{x}$ is the ith **principal component** (PC)

Case of known covariance matrix Suppose we know Σ, covariance matrix of $\mathbf{x}$ (i.e., typically: we know $\mathbf{x}$)

Then the kth PC is

$$z_k = \mathbf{m}\alpha_k^T\mathbf{x}$$

where $\mathbf{m}\alpha_k$ is an eigenvector of Σ corresponding to the kth largest eigenvalue λ_k

If, additionally, $\|\mathbf{m}\alpha_k\| = \mathbf{m}\alpha_k^T\mathbf{m}\alpha = 1$, then $\lambda_k = \mathsf{Var}\ z_k$

Why is that? Let us start with

$$\mathbf{m}\alpha_1^T\mathbf{x}$$

We want maximum variance, where $\mathbf{m}\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1p})$, i.e.,

$$\mathbf{m}\alpha_1^T\mathbf{x} = \sum_{j=1}^{p} \alpha_{1j}x_j$$

with the constraint that $\|\mathbf{m}\alpha_1\| = 1$

We have

$$\mathsf{Var}\ \mathbf{m}\alpha_1^T\mathbf{x} = \mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1$$

Objective We want to maximise $\mathsf{Var}\ \mathbf{m}\alpha_1^T\mathbf{x}$, i.e.,

$$\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1$$

under the constraint that $\|\mathbf{m}\alpha_1\| = 1$

$\Longrightarrow$ use **Lagrange multipliers**

Maximisation using Lagrange multipliers

subtitle(A.k.a. super-brief intro to multivariable calculus) We want the max of $f(x_1, \ldots, x_n)$ under the constraint $g(x_1, \ldots, x_n) = k$

1. Solve

$$\begin{aligned} \nabla f(x_1, \ldots, x_n) &= \lambda \nabla g(x_1, \ldots, x_n) \\ g(x_1, \ldots, x_n) &= k \end{aligned}$$

   where $\nabla = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n})$ is the **gradient operator**
2. Plug all solutions into $f(x_1, \ldots, x_n)$ and find maximum values (provided values exist and $\nabla g \neq \mathbf{0}$ there)

λ is the **Lagrange multiplier**

The gradient subtitle(Continuing our super-brief intro to multivariable calculus) $f : \mathbb{R}^n \to \mathbb{R}$ function of several variables, $\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)$ the gradient operator

Then

$$\nabla f = \left(\frac{\partial}{\partial x_1} f, \ldots, \frac{\partial}{\partial x_n} f\right)$$

So ∇f is a *vector-valued* function, $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$; also written as

$$\nabla f = f_{x_1}(x_1, \ldots, x_n)\mathbf{e}_1 + \cdots f_{x_n}(x_1, \ldots, x_n)\mathbf{e}_n$$

where f_{x_i} is the partial derivative of f with respect to x_i and $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ is the standard basis of $\mathbb{R}^n$

$\mathbf{m}\alpha_1^T \Sigma \mathbf{m}\alpha_1$ and $\|\mathbf{m}\alpha_1\|^2 = \mathbf{m}\alpha_1^T \mathbf{m}\alpha_1$ are functions of $\mathbf{m}\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1p})$

In the notation of the previous slide, we want the max of

$$f(\alpha_{11}, \ldots, \alpha_{1p}) := \mathbf{m}\alpha_1^T \Sigma \mathbf{m}\alpha_1$$

under the constraint that

$$g(\alpha_{11}, \ldots, \alpha_{1p}) := \mathbf{m}\alpha_1^T \mathbf{m}\alpha_1 = 1$$

and with gradient operator

$$\nabla = \left(\frac{\partial}{\partial \alpha_{11}}, \ldots, \frac{\partial}{\partial \alpha_{1p}}\right)$$

Effect of ∇ on g

g is easiest to see:

$$\begin{aligned}\nabla g(\alpha_{11},\ldots,\alpha_{1p}) &= \left(\frac{\partial}{\partial\alpha_{11}},\ldots,\frac{\partial}{\partial\alpha_{1p}}\right)(\alpha_{11},\ldots,\alpha_{1p})\begin{pmatrix}\alpha_{11}\\ \vdots\\ \alpha_{1p}\end{pmatrix}\\ &= \left(\frac{\partial}{\partial\alpha_{11}},\ldots,\frac{\partial}{\partial\alpha_{1p}}\right)\left(\alpha_{11}^2+\cdots+\alpha_{1p}^2\right)\\ &= (2\alpha_{11},\ldots,2\alpha_{1p})\\ &= 2\mathbf{m}\alpha_1\end{aligned}$$

(And that's a general result: $\nabla\|\mathbf{x}\|_2^2 = 2\mathbf{x}$ with $\|\cdot\|_2$ the Euclidean norm)

Effect of ∇ on f

Expand (write $\Sigma = [s_{ij}]$ and do not exploit symmetry)

$$\begin{aligned}\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 &= (\alpha_{11},\ldots,\alpha_{1p})\begin{pmatrix}s_{11} & s_{12} & \cdots & s_{1p}\\ s_{21} & s_{22} & \cdots & s_{2p}\\ \vdots & \vdots & & \vdots\\ s_{p1} & s_{p2} & & s_{pp}\end{pmatrix}\begin{pmatrix}\alpha_{11}\\ \alpha_{12}\\ \vdots\\ \alpha_{1p}\end{pmatrix}\\ &= (\alpha_{11},\ldots,\alpha_{1p})\begin{pmatrix}s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p}\\ s_{21}\alpha_{11}+s_{22}\alpha_{12}+\cdots+s_{2p}\alpha_{1p}\\ \vdots\\ s_{p1}\alpha_{11}+s_{p2}\alpha_{12}+\cdots+s_{pp}\alpha_{1p}\end{pmatrix}\\ &= (s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p})\alpha_{11}\\ &\quad+(s_{21}\alpha_{11}+s_{22}\alpha_{12}+\cdots+s_{2p}\alpha_{1p})\alpha_{12}\\ &\quad\vdots\\ &\quad+(s_{p1}\alpha_{11}+s_{p2}\alpha_{12}+\cdots+s_{pp}\alpha_{1p})\alpha_{1p}\end{aligned}$$

We have

$$\begin{aligned}\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 &= (s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p})\alpha_{11}\\ &\quad+(s_{21}\alpha_{11}+s_{22}\alpha_{12}+\cdots+s_{2p}\alpha_{1p})\alpha_{12}\\ &\quad\vdots\\ &\quad+(s_{p1}\alpha_{11}+s_{p2}\alpha_{12}+\cdots+s_{pp}\alpha_{1p})\alpha_{1p}\end{aligned}$$

So

$$\begin{aligned}\frac{\partial}{\partial\alpha_{11}}\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 &= (s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p})+s_{11}\alpha_{11}\\ &\quad+s_{21}\alpha_{12}\\ &\quad\vdots\\ &\quad+s_{p1}\alpha_{1p}\\ &= s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p}\\ &\quad+s_{11}\alpha_{11}+s_{21}\alpha_{12}+\cdots+s_{p1}\alpha_{1p}\\ &= 2(s_{11}\alpha_{11}+s_{12}\alpha_{12}+\cdots+s_{1p}\alpha_{1p})\end{aligned}$$

(last equality stems from symmetry of Σ)

In general, for $i=1,\ldots,p$,

$$\begin{aligned}\frac{\partial}{\partial\alpha_{1i}}\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 &= s_{i1}\alpha_{11}+s_{i2}\alpha_{12}+\cdots+s_{ip}\alpha_{1p}\\ &\quad+s_{i1}\alpha_{11}+s_{2i}\alpha_{12}+\cdots+s_{pi}\alpha_{1p}\\ &= 2(s_{i1}\alpha_{11}+s_{i2}\alpha_{12}+\cdots+s_{ip}\alpha_{1p})\end{aligned}$$

(because of symmetry of Σ)

As a consequence,

$$\nabla\mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 = 2\Sigma\mathbf{m}\alpha_1$$

So solving

$$\nabla f(x_1,\ldots,x_n) = \lambda\nabla g(x_1,\ldots,x_n)$$

means solving

$$2\Sigma\mathbf{m}\alpha_1 = \lambda 2\mathbf{m}\alpha_1$$

i.e.,

$$\Sigma\mathbf{m}\alpha_1 = \lambda\mathbf{m}\alpha_1$$

$\Longrightarrow$ $(\lambda,\mathbf{m}\alpha_1)$ eigenpair of Σ, with $\mathbf{m}\alpha_1$ having unit length

Picking the right eigenvalue

$(\lambda,\mathbf{m}\alpha_1)$ eigenpair of Σ, with $\mathbf{m}\alpha_1$ having unit length

But which λ to choose?

Recall that we want $\mathsf{Var}\ \mathbf{m}\alpha_1^T\mathbf{x} = \mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1$ maximal

We have

$$\mathsf{Var}\ \mathbf{m}\alpha_1^T\mathbf{x} = \mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_1 = \mathbf{m}\alpha_1^T(\Sigma\mathbf{m}\alpha_1) = \mathbf{m}\alpha_1^T(\lambda\mathbf{m}\alpha_1) = \lambda(\mathbf{m}\alpha_1^T\mathbf{m}\alpha_1) = \lambda$$

$\Longrightarrow$ we pick $\lambda=\lambda_1$, the largest eigenvalue (covariance matrix symmetric so eigenvalues real)

What we have this far.. The first principal component is $\mathbf{m}\alpha_1^T\mathbf{x}$ and has variance λ_1, where λ_1 the largest eigenvalue of Σ and $\mathbf{m}\alpha_1$ an associated eigenvector with $\|\mathbf{m}\alpha_1\| = 1$

We want the second principal component to be *uncorrelated* with $\mathbf{m}\alpha_1^T\mathbf{x}$ and to have maximum variance $\mathsf{Var}\ \mathbf{m}\alpha_2^T\mathbf{x} = \mathbf{m}\alpha_2^T\Sigma\mathbf{m}\alpha_2$, under the constraint that $\|\mathbf{m}\alpha_2\| = 1$

$\mathbf{m}\alpha_2^T\mathbf{x}$ uncorrelated to $\mathbf{m}\alpha_1^T\mathbf{x}$ if $\mathsf{cov}(\mathbf{m}\alpha_1^T\mathbf{x}, \mathbf{m}\alpha_2^T\mathbf{x}) = 0$

We have

$$\begin{aligned}\mathsf{cov}(\mathbf{m}\alpha_1^T\mathbf{x}, \mathbf{m}\alpha_2^T\mathbf{x}) &= \mathbf{m}\alpha_1^T\Sigma\mathbf{m}\alpha_2 \\ &= \mathbf{m}\alpha_2^T\Sigma^T\mathbf{m}\alpha_1 \\ &= \mathbf{m}\alpha_2^T\Sigma\mathbf{m}\alpha_1 \quad [\Sigma \text{ symmetric}] \\ &= \mathbf{m}\alpha_2^T(\lambda_1\mathbf{m}\alpha_1) \\ &= \lambda\mathbf{m}\alpha_2^T\mathbf{m}\alpha_1\end{aligned}$$

So $\mathbf{m}\alpha_2^T\mathbf{x}$ uncorrelated to $\mathbf{m}\alpha_1^T\mathbf{x}$ if $\mathbf{m}\alpha_1 \perp \mathbf{m}\alpha_2$

This is beginning to sound a lot like Gram-Schmidt, no?

In short Take whatever covariance matrix is available to you (known Σ or sample S_X) – assume sample from now on for simplicity

For $i = 1, \ldots, p$, the ith principal component is

$$z_i = \mathbf{v}_i^T\mathbf{x}$$

where $\mathbf{v}_i$ eigenvector of S_X associated to the ith largest eigenvalue λ_i

If $\mathbf{v}_i$ is normalised, then $\lambda_i = \mathsf{Var}\ z_k$

Covariance matrix Σ the covariance matrix of the random variable, S_X the sample covariance matrix

$X \in \mathcal{M}_{mp}$ the data, then the (sample) covariance matrix S_X takes the form

$$S_X = \frac{1}{n-1}X^TX$$

where the data is centred!

Sometimes you will see $S_X = 1/(n-1)XX^T$. This is for matrices with observations in columns and variables in rows. Just remember that you want the covariance matrix to have size the number of variables, not observations, this will give you the order in which to take the product

5.1.1 Hockey players (eh!)

Let us consider a very Canadian example, although it involves players from all over the world. The height and weight of hockey players who participated in IIHF world championship games during the period 2001-2016 can be found online (here), as part of a study.

We download the data

```
> #data = read.csv("https://figshare.com/ndownloader/files/5303173")
> data = read.csv("DATA/hockey_players.csv")
```

and take a quick peek (using `kable` and `head`), focusing on the first few columns.

year	country	no	name	position	side	height	weight	birth
2001	RUS	10	tverdovsky oleg	D	L	185	84	1976-05-18
2001	RUS	2	vichnevsky vitali	D	L	188	86	1980-03-18
2001	RUS	26	petrochinin evgeni	D	L	182	95	1976-02-07
2001	RUS	28	zhdan alexander	D	R	178	85	1971-08-28
2001	RUS	32	orekhovsky oleg	D	R	175	88	1977-11-03
2001	RUS	4	zhukov sergei	D	L	193	93	1975-11-23

There is also information about players' club, age, age cohort and BMI, but we will not be using any of these for sure. The table contains 6292 rows. However, the author of the study was interested in the evolution of weights, so it is likely that the same person will be in the dataset several times. Let us check if this is indeed the case. The command `any` returns true if any of the entries in the vector it is passed as an argument is true and `duplicated` is true if an entry is present more than once in its argument.

```
> any(duplicated(data$name))
```

```
[1] TRUE
```

So, indeed, there are players present several times. We are not interested in the evolution of weights, so let us simplify things: if we have more than one record for someone, let us take their height and weight as the average of the heights and weights recorded for them. We keep only three characteristics for a player: their country, height and weight. (We do keep their name as well.) It could be worth checking if there is any case of a player playing for two different countries, which would require to decide on a mechanism to attribute a country to them. However, in this illustrative example, we choose to ignore this possibility.

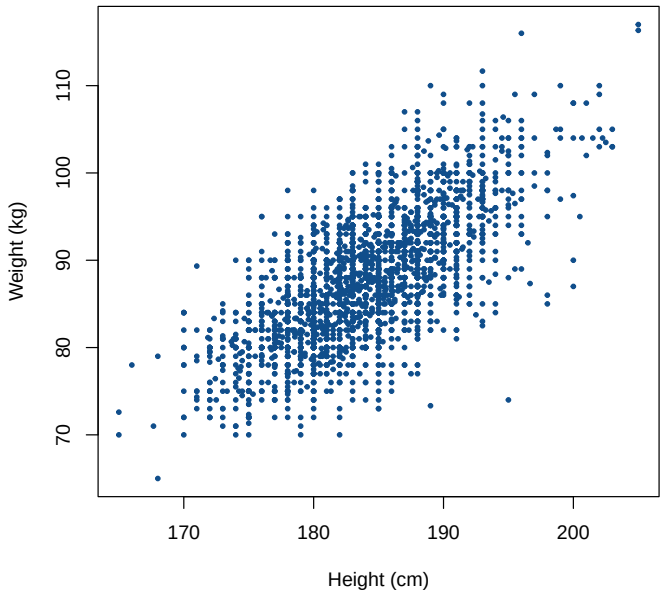
```
> data_simplified = data.frame(name = unique(data$name))
> c = c()
> w = c()
> h = c()
> for (n in data_simplified$name) {
+     tmp = data[which(data$name == n),]
+     c = c(c, tmp$country[1])
+     h = c(h, mean(tmp$height))
+     w = c(w, mean(tmp$weight))
+ }
> data_simplified$country = c
```

```
> data_simplified$weight = w
> data_simplified$height = h
> data = data_simplified
```

The resulting table looks like this

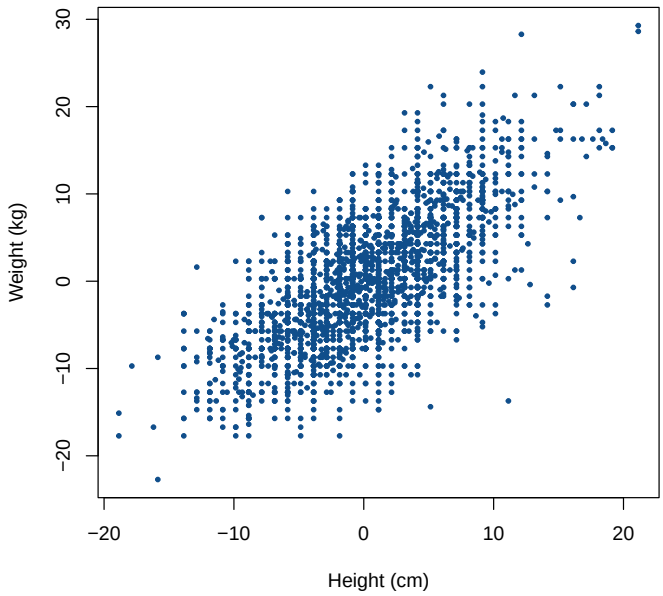
name	country	weight	height
tverdovsky oleg	RUS	84.0	185.0
vichnevsky vitali	RUS	86.0	188.0
petrochinin evgeni	RUS	95.0	182.0
zhdan alexander	RUS	85.5	178.5
orekhovsky oleg	RUS	88.0	175.0
zhukov sergei	RUS	92.5	193.0

Let us plot the height and weight data as it stands after this preprocessing step.



The mean height of players in the data is 183.9 centimetres, their mean weight is 87.72 kilograms. Let us centre the data:

```
> data$weight.c = data$weight-mean(data$weight)
> data$height.c = data$height-mean(data$height)
```



If you do not pay attention to the axes, you can see no difference between the two figures. Although this is a change of basis, this is a rather simple one. Let us now compute the covariance of the data.

```
> cov(data$height, data$weight)
```

```
[1] 26.63506
```

(Note that using the centred data would have given exactly the same result.) So there is a positive linear relationship between the two variables (duh!). Let us now compute the sample covariance matrix, using the centred data.

```
> X = as.matrix(data[,c("height.c", "weight.c")])
> S = 1/(dim(X)[1]-1)*t(X) %*% X
```

The result is a 2×2 matrix,

```
> S
```

```
          height.c weight.c
height.c 29.66176 26.63506
weight.c 26.63506 47.81112
```

One important remark here: the sample covariance matrix, when computed using this method, *requires* the data to be centred. If you do not centre the data prior to computing that matrix, this is what you find.

```
> X2 = as.matrix(data[,c("height", "weight")])
> S2 = 1/(dim(X)[1]-1)*t(X2) %*% X2
> S2
```

```
         height    weight
height 33844.33 16158.902
weight 16158.90  7744.176
```

Let us now compute the principal components. For this, we need eigenvalues and eigenvectors.

```
> ev = eigen(S)
> ev
```

```
eigen() decomposition
$values
[1] 66.87496 10.59793

$vectors
          [,1]       [,2]
[1,] 0.5820222 -0.8131729
[2,] 0.8131729  0.5820222
```

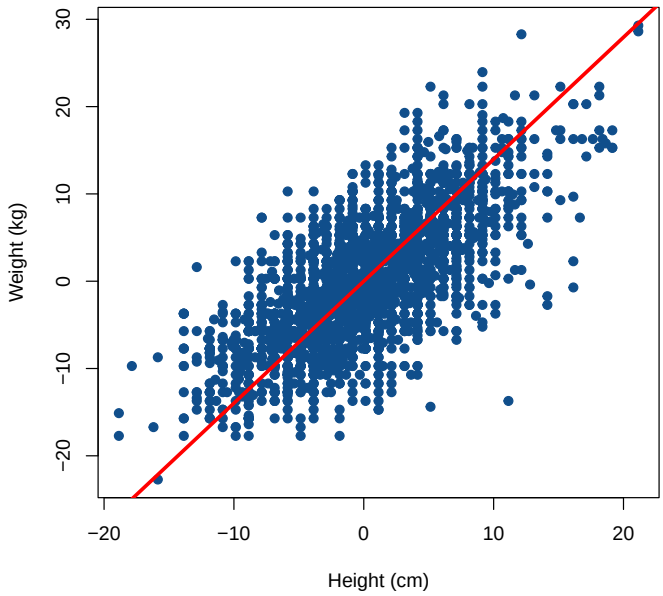
For matrices with only real eigenvalues, `eigen` returns eigenvalues sorted in decreasing order. If we needed to ensure that eigenvalues (and their corresponding eigenvectors) are ordered as we want them to be, we would for instance proceed as follows.

```
> idx_order = order(ev$values, decreasing = TRUE)
> ev$values = ev$values[idx_order]
> ev$vectors = ev$vectors[, idx_order]
```

Let us normalise the first eigenvector. This way, we now that the variance of the first principal component is the corresponding eigenvalue.

```
> ev$vectors[,1] = ev$vectors[,1] / sqrt(sum(ev$vectors[,1]^2))
```

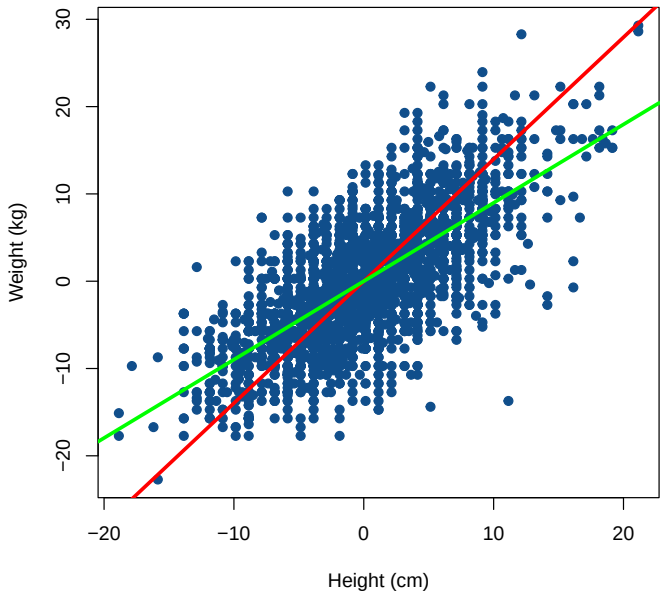
Let us plot this first eigenvector (well, the line carrying this first eigenvector). To use the function 'abline', we need to give the coefficients of the line in the form of (intercept,slope). Intercept is easy, as the line goes through the origin (by construction and because we have centred the data). The slope is also quite simple..



I will let you in a little "secret": least squares are used so often in Stats that 'R' has a very simple function that does that very well. The function is 'lm' (linear models) and you can use it as follows.

```
> z <- lm(weight.c ~ 0 + height.c, data = data)
```

This essentially says "fit a linear model with `weight.c` function of `height.c` and store the result in `z`". You can then use this result in a variety of contexts. First of all, this is what the result looks like (in green), in comparison to the one we found.



Why, you may ask, are the two results so different?

Let us rotate the data so that the red line becomes the red axis. To do that, we use a rotation matrix,

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

To find the angle θ, recall that $\tan\theta$ is equal to opposite length over adjacent length, i.e.,

$$\tan\theta = \frac{\texttt{ev\$vectors}[2,1]}{\texttt{ev\$vectors}[1,1]}$$

So we just use the arctan of this. (Note that angles are in radians.)

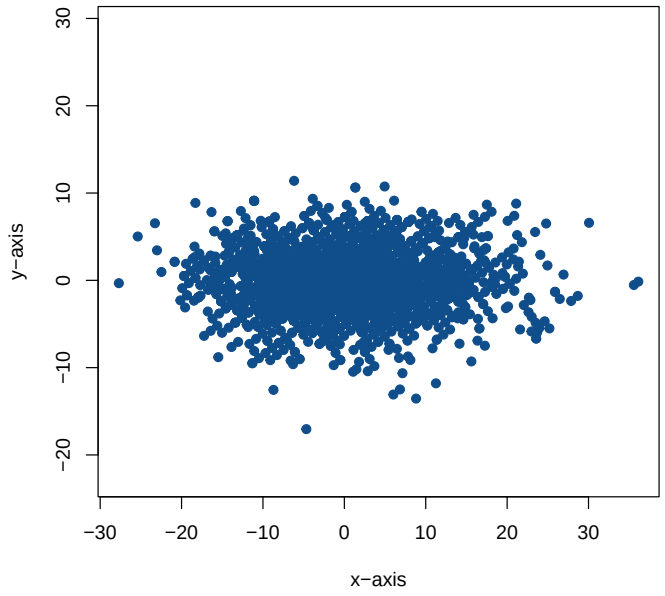
```
> theta = atan(ev$vectors[2,1]/ev$vectors[1,1])
> R_theta = matrix(c(cos(theta), -sin(theta),
+                    sin(theta), cos(theta)),
+                  nr = 2, byrow = TRUE)
```

And now we rotate the points. (In this case, we think of the points as vectors, of course.)

```
> tmp_in = matrix(c(data$weight.c, data$height.c),
+                 nc = 2)
> tmp_out = c()
> for (i in 1:dim(tmp_in)[1]) {
+     tmp_out = rbind(tmp_out,
```

```
+                       t(R_theta %*% tmp_in[i,]))
+ }
> data$weight.c_r = tmp_out[,1]
> data$height.c_r = tmp_out[,2]
```

This change of basis has recast the data in a way where the first axis (the x-axis) is the dimension along which there is the highest variance. To see this, we need to plot the data with the same range as was used before.



It is of course possible to do the same thing with existing R functions.

```
> if (!require("pracma")) {
+     install.packages("pracma")
+ }
> GS = pracma::gramSchmidt(A = ev$vectors)
```

Now recall we saw a theorem that told us how to construct a new basis.

```
> A=matrix(c(GS$Q,1,0,0,1), nr = 2)
> pracma::rref(A)
```

```
     [,1] [,2]       [,3]      [,4]
[1,]    1    0  0.5820222 0.8131729
[2,]    0    1 -0.8131729 0.5820222
```

```
> P = pracma::rref(A)[,c(3,4)]
> X.new = X %*% t(P)
```

Chapter 6

Graph theory ... theory

In this chapter, I present some elements of graph theory. Chapters that follow are devoted to applications, but here I focus on the theory. I will nonetheless explain how you can use `R` to perform some of the operations or characterise some of the structures described in this chapter.

6.1 Introduction and preliminaries

6.1.1 Graphs versus networks

First, let me remark that it is likely that you know some of the material here under the name of *network theory*. This is mostly a terminology difference:

- we say *graphs* in the mathematical world;
- most of the rest of the world says *networks*.

I will mostly say *graphs* as this is a mathematics course, but might oscillate. Beware: this is a domain of mathematics in which the language is not consistent, even more so because of the duality noted above. Make sure you read the definitions at the start of whatever source you are using.

6.1.2 Graphs vs digraphs vs multigraphs vs multidigraphs vs ...

As noted, name-wise and notation-wise, this domain is a bit of a mess. We see the precise definitions of the objects below later, but to clarify notation, let me highlight notation choices here.

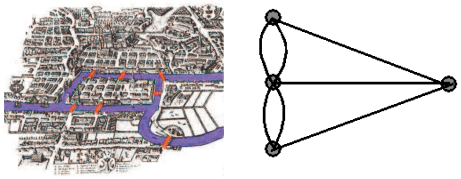
- The *vertex set* V is essentially the only constant in what follows.
- An *undirected graph* is denoted $G = (V, E)$, where E are the *edge*s.
- An *undirected multigraph* is denoted $G_M = (V, E)$. We will not be using these much.
- A *directed graph* (or *digraph*) is denoted $G = (V, A)$, where A are the *arcs*.
- A *directed multigraph* (or *multidigraph*) is denoted $G_M = (V, A)$.

- Any of the above is called a graph and is denoted $G = (V, X)$, when we seek generality.

And just to confuse the whole thing more: we often say *graph* for *unoriented graph*.

6.1.3 The bridges of Königsberg

A "real life" problem formulated by Euler: is it possible to cross the seven bridges of Königsberg in a continuous walk without recrossing any of them?



Mathematical problem Is it possible to find a trail containing all edges of the graph?

6.1.4 Finding a cycle with all vertices

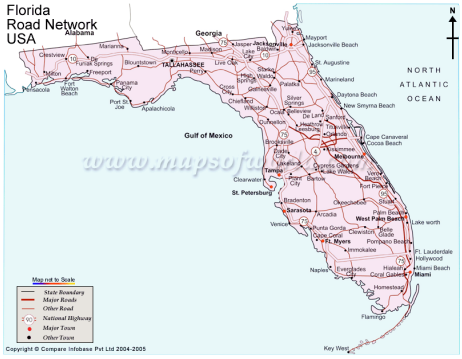
A salesperson must visit a couple of cities for their job. Is it possible for them to plan a round trip using highways enabling him to visit each specified city exactly once?

- vertices correspond to cities
- two vertices are connected iff a highway connects the corresponding cities and does not pass through any other city.

Mathematical problem Is it possible to find a cycle containing all graph vertices?

6.1.5 How far is it to drive through n cities?

What is the minimal length of driving needed to drive through n cities?



- vertices correspond to the cities
- all cities are connected; each edge has a value assigned to it

  Mathematical problem What is the minimal spanning tree associated to the graph?

6.2 Binary relations

Graphs are used to describe relations between elements (the vertices or nodes). Mathematically, relations can be encoded by *binary relations*, which we define here.

Definition 6.1.
- *A **binary relation** is an arbitrary association of elements of one set with elements of another (maybe the same) set.*
- *A binary relation over the sets X and Y is a subset of the Cartesian product $X \times Y = \{(x, y) | x \in X, y \in Y\}$.*
- *$(x, y) \in R$ is read "x is $R-$related to y" and is denoted xRy.*
- *If $(x, y) \notin R$, we write not xRy.*

Definition 6.2 (Properties of binary relations)**.** *A binary relation R over a set X is*
- ***Reflexive** if $\forall x \in X$, xRx.*
- ***Irreflexive** if there does not exist $x \in X$ such that xRx.*
- ***Symmetric** if $xRy \Rightarrow yRx$.*
- ***Asymmetric** if $xRy \Rightarrow$ not yRx.*
- ***Antisymmetric** if xRy and $yRx \Rightarrow x = y$.*
- ***Transitive** if xRy and $yRz \Rightarrow xRz$.*
- ***Total** (or **complete**) if $\forall x, y \in X$, xRy or yRx.*

Definition 6.3 (Equivalence relation)**.** *A relation which is reflexive, symmetric and transitive is called an **equivalence relation**.*

An equivalence relation allows to partition the set of elements on which it is defined into **equivalence classes**. In a given equivalence class, all elements are equivalent and any element can be chosen to **represent** that equivalence class. If S is the set over which the relation R is an equivalence relation, we denote S/R the **quotient set**.

Definition 6.4 (Partial order)**.** *A relation which is reflexive, antisymmetric and transitive is called a **partial order**.*

6.3 Undirected graphs

6.3.1 Undirected graph

Intuitively: a graph is a set of points, and a set of relations between the points. The points are called the *vertices* of the graph and the relations are the *edges* of the graph. We can also think of the relations as being one directional, in which case the relations are the *arcs* of the digraph (a contraction of "directed graph").

Definition 6.5 (Graph)**.** *An **undirected graph** is a pair $G = (V, E)$ of sets such that*

- *V is a set of points: $V = \{v_1, v_2, v_3, .., v_p\}$*
- *E is a set of 2-element subsets of V: $E = \{\{v_i, v_j\}, \{v_i, v_k\}, \ldots, \{v_n, v_p\}\}$, also noted $E = \{v_iv_j, v_iv_k, \ldots, v_nv_p\}$.*

Definition 6.6 (Vertex)**.** *The elements of V are the **vertices** (or nodes, or points) of the graph G. V is the vertex set of the graph G, also noted $V(G)$.*

Definition 6.7 (Edge)**.** *The elements of E are the **edges** (or lines) of the graph G. E is the edge set of the graph G, also noted $E(G)$.*

In `R`, we will use mostly the package `igraph`, which has the advantage of also existing in `Python`.

```
> if (!require(igraph)) {
+   install.package("igraph")
+ }
```

There are many different ways to construct graphs in `igraph`. There are also means to construct graphs with specific properties. I do not detail all of them here. Here are a few of the methods for creating "your own" graphs.

- `graph_from_adjacency_matrix` creates graphs from adjacency matrices; see Section 6.6.1.
- `graph_from_adj_list` creates graphs from adjacency lists.
- `graph_from_edgelist` creates a graph from an edge list matrix.
- `graph_from_incidence_matrix` creates graphs from incidence matrices.
- `graph_from_literal` creates (small) graphs via a simple interface.

There are many other methods, typically with a name starting with the prefix `graph_from`, see the list of `igraph` functions. Let us create a simple graph from an edge list. If we have, say, a graph with 6 vertices labelled a through f, we could for instance create the matrix

```
> M = matrix(c("a", "b",
+              "b", "c",
+              "b", "d",
+              "c", "d",
```

```
+                "c", "f",
+                "e", "f",
+                "e", "a"),
+              nc = 2, byrow = TRUE)
```

We now use the function `graph_from_edgelist` to make the graph. Note that this function assumes by default that the graph is *oriented*, i.e., a digraph, so to make an undirected graph, we use the optional argument `directed = FALSE`.

```
> G = graph_from_edgelist(M, directed = FALSE)
```

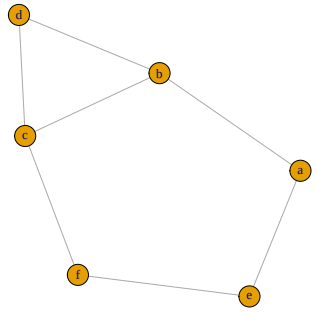
You can get some basic information about the graph you just created by typing its name.

```
> G
```

```
IGRAPH 7ebce05 UN-- 6 7 --
+ attr: name (v/c)
+ edges from 7ebce05 (vertex names):
[1] a--b b--c b--d c--d c--f f--e a--e
```

It is also easy to plot the graph, although by default, the result is not really impressive.

```
> plot(G)
```



6.3.2 Order and size of graph

Two fundamental properties of a graph are the number of vertices and of edges it has. These numbers are called, respectively, the order and size of the graph.

Definition 6.8 (Order of a graph)**.** *The number of vertices in G is the* ***order*** *of G. Using the notation $|V(G)|$ for the* cardinality *of $V(G)$,*

$$|V(G)| = \textit{order of } G.$$

Definition 6.9 (Size of a graph)**.** *The number of edges in G is the **size** of G,*

$$|E(G)| = \textit{size of } G.$$

- A graph having order p and size q is called a $(p, q)-$graph.
- A graph is **finite** if $|V(G)| < \infty$.

These properties are obtained in `igraph` using

```
> gorder(G)

[1] 6

> gsize(G)

[1] 7
```

6.3.3 Relationships between vertices and edges, nature of the edges

Definition 6.10 (Incident vertices and edges)**.** *Let $G = (V, E)$ be an undirected graph.*

- *A vertex v is incident with an edge e if $v \in e$; then e is an edge at v.*
- *If $e = uv \in E(G)$, then u and v are each incident with e.*
- *Two vertices incident with an edge are its ends.*
- *An edge $e = uv$ is incident with both vertices u and v*

Definition 6.11 (Adjacent)**.**

- *Two vertices u and v are **adjacent** in a graph G if $uv \in E(G)$.*
- *If uv and uw are distinct edges (i.e. $v \neq w$) of a graph G, then uv and uw are adjacent edges.*

Definition 6.12 (Multiple edge)**. Multiple edges** *are two or more edges connecting the same two vertices within a multigraph.*

Definition 6.13 (Loop)**.** *A* **loop** *is an edge with both the same ends;* e.g. $\{u, u\}$ *is a loop.*

Definition 6.14 (Simple graph)**.** *A* **simple graph** *is a graph which contains no loops or multiple edges.*

Definition 6.15 (Multigraph)**.** *A* **multigraph** *is a graph which can contain multiple edges or loops.*

Several `igraph` functions allow to consider these aspects. From the documentation,

- `any_loop` decides whether the graph has any loop edges.
- `which_loop` decides whether the edges of the graph are loop edges.
- `any_multiple` decides whether the graph has any multiple edges.
- `which_multiple` decides whether the edges of the graph are multiple edges.
- `count_multiple` counts the multiplicity of each edge of a graph.

Graph and binary relations A (simple) graph G can be defined in term of a vertex set V and an irreflexive and symmetric binary relation over V. R is symmetric if $(u, v) \in R \Rightarrow (v, u) \in R$ (or, in other words, $uRv \implies vRu$). Hence, $\{(u, v), (v, u)\} \in E(G)$ ($\{(u, v), (v, u)\}$ is an edge). The set of edges $E(G)$ is the set of symmetric pairs in R.

6.3.4 Degree of a vertex

Definition 6.16 (Degree of a vertex)**.** *Let v be a vertex of $G = (V, E)$.*

- *The number of edges of G incident with v is the* **degree** *of v in G.*
- *The number of edges of G at v is the* **degree** *of v in G.*
- *The degree of v in G is noted $d_G(v)$ or $deg_G(v)$.*

Theorem 6.17. *Let G be a $(p, q)-$graph with vertices $v_1, \ldots, v_p$, then*

$$\sum_{i=1}^{p} d_G(v_i) = 2q.$$

Definition 6.18 (Odd vertex)**.** *A vertex is an* ***odd vertex*** *is its degree is odd.*

Definition 6.19 (Even vertex)**.** *A vertex is called* ***even vertex*** *is its degree is even.*

Theorem 6.20. *Every graph contains an even number of odd vertices.*

`igraph` has many functions to consider degrees, as they are important to characterise the properties of graphs. Some are detailed in Chapter 7, here we just give the base. To obtain the degree of vertex e in the graph defined above,

```
> degree(G, "e")


e
2
```

(`degree(G,c("a","e"))` would return the degree of a and e), while

```
> degree(G)


a b c d f e
2 3 3 2 2 2
```

returns the degree of all vertices in G.

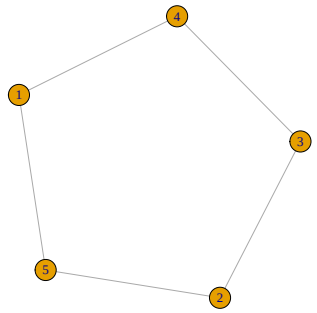
6.3.5 Regular, complete, bipartite and other notable graphs

Some graphs are very useful in studying the mathematical properties of graphs or in applications and have therefore been named. I list some of these below. Note that `igraph` has many functions to generate such graphs as well, so most graphs below are also illustrated using `igraph`. Typically, if a function exists to create a graph with prescribed properties, a function also exists to check if an existing graph possesses that property. I also indicate these functions.

Definition 6.21 (Regular graph)**.** *If all the vertices of G have the same degree k, then the graph G is $k-$regular.*

The command below generates a random regular undirected graph of order 5 where each vertex has degree 2.

```
> U = sample_k_regular(no.of.nodes = 5, k = 2)
```



Note that it is "easy" to specify values of the order and degree for which no graph can be found. In this case, you will get an error. There is no function to check if a graph is regular, although this is very easily done using `degree`. For instance,

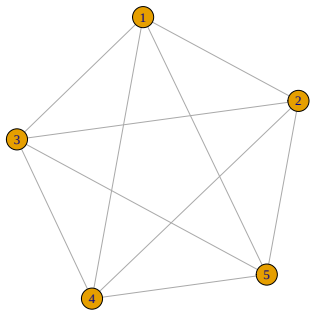
```
> length(unique(degree(G))) == 1

[1] FALSE

> length(unique(degree(U))) == 1

[1] TRUE
```

Definition 6.22 (Complete graph)**.** *A graph is complete if every two of its vertices are adjacent.*

Definition 6.23 (n-clique)**.** *A simple, complete graph on n vertices is called an n-**clique** and is often denoted K_n.*

Note that a complete graph of order p is $(p-1)-$regular. In `igraph`, complete graphs are called *full* graphs, so the function to create them are as follows.

```
> U = make_full_graph(5)
```

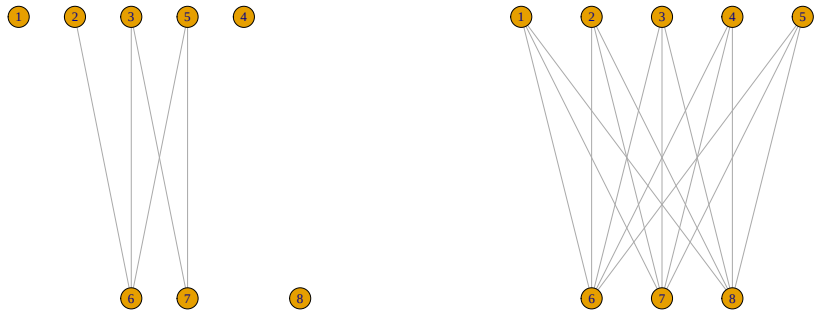


Definition 6.24 (Bipartite graph)**.** *A graph $G = (V, E)$ is **bipartite** if it is possible to partition the vertex set $V(G)$ into two subsets V_1 and V_2 so that every edge of G joins a vertex of V_1 with a vertex of V_2 and no vertex joins another vertex of its own set, i.e., no two vertices in the same set are adjacent. This graph may be written $G = (X_1, X_2, U)$*

Definition 6.25 (Complete bipartite graph)**.** *A bipartite graph in which every two vertices from the 2 different partitions are adjacent is called a **complete bipartite graph**. A simple, complete bipartite graph with $|X_1| = p$ and $|X_2| = q$ is often denoted $K_{p,q}$.*

Bipartite graphs can be generated in `igraph` using several different commands. If you know the specific connections, you can use `make_bipartite_graph`. Otherwise, a random bipartite graph with specified number of vertices in the two sets is defined using `sample_bipartite`, which we do here. Note that the argument `p = 0.3` is the probability that an edge exists. Complete bipartite graphs are generated using `make_full_bipartite_graph`.

```
> U1 = sample_bipartite(n1 = 5, n2 = 3, p = 0.3)
> U2 = make_full_bipartite_graph(n1 = 5, n2 = 3)
```

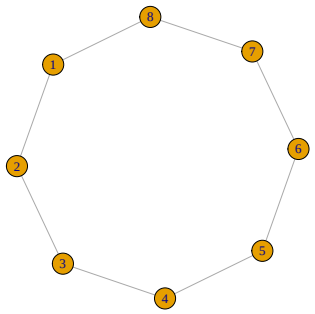


Note that the plots above use an additional option for the vertices to appear as they do. For instance, the graph on the right was obtained using `plot(U2, layout = layout_bipartite)`. You can also easily check if a graph is bipartite using the command `is_bipartite`, so for instance

```
> is_bipartite(U)

[1] FALSE

> is_bipartite(U1)

[1] TRUE
```

Definition 6.26 (Cycle C_n)**.** *For $n \geq 3$, the cycle, denoted C_n, is a connected graph of order n that is a cycle on n vertices.*

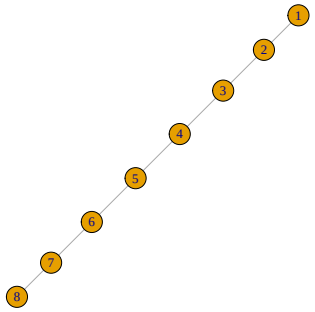
```
> U = make_ring(n = 8)
```



Definition 6.27 (Path P_n)**.** *The path, P_n, is a connected graph that consists of $n \geq 2$ vertices and $n-1$ edges. Two vertices of P_n have degree 1 and the rest are of degree 2.*

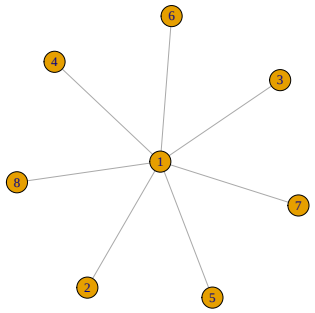
Note that to obtain a path in `igraph`, we use the same function as for cycles but with an additional argument.

```
> U = make_ring(n = 8, circular = FALSE)
```



Definition 6.28 (Star S_n)**.** *The star of order n is the complete bipartite graph $K_{1,n-1}$ (1 vertex of degree $n-1$, and $n-1$ vertices of degree 1).*

```
> U = make_star(n = 8, mode = "undirected")
```



6.3.6 Isomorphic graphs

Definition 6.29 (Isomorphic graphs)**.** *Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs. G_1 and G_2 are **isomorphic** if there exists an isomorphism ϕ from G_1 to*

G_2, that is defined as an injective mapping $\phi : V(G_1) \to V(G_2)$ such that two vertices u_1 and v_1 are adjacent in G_1 if and only if the vertices $\phi(u_1)$ and $\phi(v_1)$ are adjacent in G_2.

Definition 6.30 (Another formulation of isomorphic graphs)**.** *Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs; G_1 and G_2 are isomorphic if there exists an isomorphism ϕ from G_1 to G_2, defined as an injective mapping $\phi : V(G_1) \to V(G_2)$ such that $\{u_1, v_1\} \in E(G_1)$ if and only if $\{\phi(u_1), \phi(v_1)\} \in E(G_2)$ for all $u_1, v_1 \in V(G_1)$.*

If ϕ is an isomorphism from G_1 to G_2, then the inverse mapping ϕ^{-1} from $V(G_2)$ to $V(G_1)$ also satisfies the definition of an isomorphism.

As a consequence, if G_1 and G_2 are isomorphic graphs, then

- G_1 is isomorphic to G_2
- G_2 is isomorphic to G_1

Theorem 6.31. *The relation "is isomorphic to" is an equivalence relation on the set of all graphs.*

Theorem 6.32. *If G_1 and G_2 are isomorphic graphs, then the degrees of vertices of G_1 are exactly the degrees of vertices of G_2.*

As you can probably guess, this is not an easy problem from an algorithmic point of view. In `igraph`, there is an exact method only if the two graphs have three or four vertices. Otherwise, the methods used can be costly from a computational point of view. The example below is adapted from the `igraph` manual: we make a graph (here, Erdos-Renyi) and permutate the vertices, giving two graphs that are isomorphic by construction.

```
> g1 <- sample_gnp(n = 50, p = 0.5)
> g2 <- permute(g1, sample(gorder(g1)))
> isomorphic(g1, g2, method = "auto")

[1] TRUE
```

6.3.7 Subgraphs, unions of graphs

Definition 6.33 (Subgraph)**.** *Let $G = (V(G), E(G))$ be a graph. A graph $H = (V(H), E(H))$ is a **subgraph** of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.*

Definition 6.34 (Another definition of subgraph)**.** *If a graph F is isomorphic to a subgraph H of G, then F is also called a subgraph of G.*

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs.

Definition 6.35 (Union of G_1 and G_2)**.** $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$

Definition 6.36 (Intersection of G_1 and G_2)**.** $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$

Definition 6.37 (Disjoint graphs)**.** *If $G_1 \cap G_2 = (\emptyset, \emptyset) = \emptyset$ (empty graph) then G_1 and G_2 are disjoint.*

Definition 6.38 (Complement of G_1)**.** *The complement $\bar{G}_1$ of G_1 is the graph on V_1, with the edge set $E(\bar{G}_1) = [V_1]^2 \backslash E_1$ ($e \in E(\bar{G}_1)$ iff $e \notin E_1$).*

6.3.8 Walks, trails, paths

Definition 6.39 (Walk)**.** *A **walk** in a graph $G = (V, E)$ is a non-empty alternating sequence $v_0 e_0 v_1 e_1 v_2 \dots e_{k-1} v_k$ of vertices and edges in G such that $e_i = \{v_i, v_{i+1}\}$ for all $i < k$. This walk begins with v_0 and ends with v_k.*

Definition 6.40 (Length of a walk)**.** *The length of a walk is equal to the number of edges in the walk.*

Definition 6.41 (Closed walk)**.** *If $v_0 = v_k$, the walk is **closed**.*

Definition 6.42 (Trail)**.** *A **trail** in G is a walk in G in which all edges are distinct.*

Definition 6.43 (Path)**.** *A **path** in G is a walk in G in which all vertices are distinct.*

The sets of vertices and edges determined by a trail or a path is a subgraph of G. Having defined paths, a question comes naturally: what is the length of a path. Clearly, one can count vertices or edges along the path. Now, given two vertices, there may be several paths joining them. To account for this, we define the distance between vertices as the length of the shortest path joining the vertices.

Definition 6.44 (Distance between two vertices)**.** *The distance $d(u, v)$ in G between two vertices u and v is the length of the shortest path linking u and v in G. If no such path exists, we assume $d(u, v) = \infty$.*

We return to the notion of distance between vertices in more detail in Chapter 7, where we will restate this definition using the name *geodesic distance* instead of just *distance* as we did here. Indeed, when characterising the graphs as we do there, it is important to consider that the edges may have a value and that the distance between edges could use this value. We defer consideration of programmatic aspects until Chapter 7.

Definition 6.45 (Circuit)**.** *A trail linking u to v, containing at least 3 edges and in which $u = v$, is a **circuit**.*

Definition 6.46 (Cycle)**.** *A circuit which does not repeat any vertices (except the first and the last) is a **cycle** (or **simple circuit**).*

Definition 6.47 (Length of a cycle)**.** *The **length of a cycle** is its number of edges.*

6.3.9 Eulerian graphs

Here, we return to one of the problems mentioned in the introduction of this chapter.

Definition 6.48 (Eulerian trail)**.** *A trail containing all the vertices and edges of a multigraph M is called a Eulerian trail of M.*

Definition 6.49 (Traversable graph)**.** *If a graph G has a Eulerian trail, then G is called a* ***traversable graph****.*

Definition 6.50 (Eulerian circuit)**.** *A circuit containing all the vertices and edges of a multigraph M is called a Eulerian circuit of M.*

Definition 6.51 (Eulerian graph)**.** *A graph (resp. multigraph) containing an Eulerian circuit is called a Eulerian graph (resp. multigraph).*

Theorem 6.52. *A multigraph M is traversable if and only if M is connected and has exactly two odd vertices. Furthermore, any Eulerian trail of M begins at one of the odd vertices and ends at the other odd vertex.*

Theorem 6.53. *A multigraph M is Eulerian if and only if M is connected and every vertex of M is even.*

Fleury's algorithm to find a Eulerian circuit (for a connected graph with no odd vertices)

- Pick any vertex as a starting point.
- Marking your path as you move from vertex to vertex, travel along any edges you wish, but DO NOT travel along an edge that is a bridge for the graph formed by the EDGES THAT HAVE YET TO BE TRAVELED – unless you have to.
- Continue until you return to your starting point.

RESULT: a Eulerian circuit

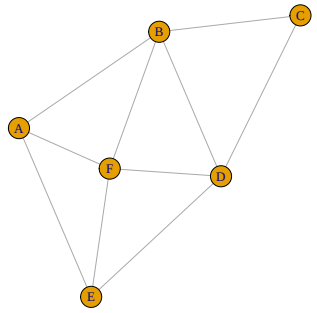
Fleury's algorithm to find a Eulerian trail (for a connected graph with exactly 2 odd vertices)

- Start at one of the odd vertices.
- Marking your path as you move from vertex to vertex, travel along any edges you wish, but DO NOT travel along an edge that is a bridge for the graph formed by the EDGES THAT HAVE YET TO BE TRAVELED – unless you have to.
- Continue until every edge has been travelled.

RESULT: a Eulerian trail

`igraph` has several functions for dealing with Eulerian graphs: `has_eulerian_path` and `has_eulerian_cycle` decide if a graph has, respectively, a Eulerian path and cycle, while `eulerian_path` and `eulerian_cycle` find Eulerian paths and cycles, respectively. The following is slightly adapted from the `igraph` help. First, we create a graph using a different method from the ones we have used this far.

```
> g <- make_graph( ~ A-B-C-D-E-A-F-D-B-F-E )
```




```
> if (has_eulerian_path(g)) {
+   eulerian_path(g)
+ } else {
+   writeLines("No Eulerian path")
+ }

$epath
+ 10/10 edges from c4d5214 (vertex names):
 [1] A--B B--C C--D B--D B--F A--F A--E D--E D--F E--F

$vpath
+ 11/6 vertices, named, from c4d5214:
 [1] A B C D B F A E D F E

> if (has_eulerian_cycle(g)) {
+   eulerian_cycle(g)
+ } else {
+   writeLines("No Eulerian cycle")
+ }

No Eulerian cycle
```

The return values are the edges along the Eulerian path or cycle and the vertices along the path or cycle. In this example, there is a Eulerian path but no Eulerian cycle.

6.3.10 Hamiltonian graphs

Definition 6.54 (Hamiltonian path)**.** *A path containing all vertices of a graph G is called a* ***Hamiltonian path*** *of G.*

Definition 6.55 (Traceable graph)**.** *If a graph G has an Hamiltonian path, then G is called a* ***traceable graph****.*

Definition 6.56 (Hamiltonian cycle)**.** *A cycle containing all vertices of a graph G is called a* ***Hamiltonian cycle*** *of G.*

Definition 6.57 (Hamiltonian graph)**.** *A graph containing a Hamiltonian cycle is called a* ***Hamiltonian graph****.*

Theorem 6.58 (Dirac's theorem)**.** *If G is a graph of order $p \geq 3$ such that $deg(v) \geq p/2$ for every vertex v of G, then G is Hamiltonian.*

Theorem 6.59 (Ore's theorem)**.** *If G is a graph of order $p \geq 3$ such that for all distinct nonadjacent vertices u and v of G,*

$$deg(u) + deg(v) \geq p,$$

then G is Hamiltonian.

6.3.11 Connectedness

Definition 6.60 (Connected vertices)**.** *Two vertices u and v in a graph G are* ***connected*** *if $u = v$, or if $u \neq v$ and there exists a path in G that links u and v.*

Definition 6.61 (Connected graph)**.** *A graph is* ***connected*** *if every two vertices of G are connected; otherwise, G is* ***disconnected****.*

To check for connectedness in `igraph`, you can use the function `is_connected`. Using the graph defined earlier in Section 6.3.9,

```
> is_connected(g)

[1] TRUE
```

Theorem 6.62 (A necessary condition for connectedness)**.** *A connected graph on p vertices has at least $p - 1$ edges.*

In other words, a connected graph G of order p has size$(G) \geq p - 1$.

Connectedness is an equivalence relation Denote $x \equiv y$ the relation "$x = y$, or $x \neq y$ and there exists a path in G connecting x and y". $\equiv$ is an equivalence relation since

1. $x \equiv y$ [reflexivity]
2. $x \equiv y \implies y \equiv x$ [symmetry]
3. $x \equiv y, y \equiv z \implies x \equiv z$ [transitivity]

Definition 6.63 (Connected component of a graph)**.** *The classes of the equivalence relation $\equiv$ partition X into connected sub-graphs of G called* ***connected components*** *(or* ***components*** *for short) of G.*

A connected subgraph H of a graph G is a component of G if H is not contained in any connected subgraph of G having more vertices or edges than H.

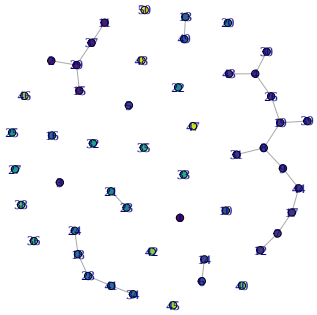
To obtain the connected components, use the function `components`. Let us create a disconnected graph, e.g., by making an Erdos-Renyi graph with low connection probability of the vertices.

```
> g <- sample_gnp(50, 1/50)
> components(g)
```

```
$membership
 [1]  1  2  3  4  5  6  4  4  4  7  2  4  8  6  2  9  4 10  4 11 12 13 12 10 14
[26]  4 15 10  2  4  4 16 17 10 18 19  2 20  4 21 10 22  4  4 23 24 25 26  8 27

$csize
 [1]  1  5  1 13  1  2  1  2  1  5  1  2  1  1  1  1  1  1  1  1  1  1  1  1  1
[26]  1  1

$no
[1] 27
```

The return value `no` is evidently the number of connected components, `membership` indicates which component a given vertex belongs to (vertices being listed in order) and `csize` is the size of the components. Here, there are 27 components. We can colour the vertices to indicate what components they belong to.

```
> comps <- components(g)$membership
> colbar = viridis::viridis(max(comps)+1)
> V(g)$color <- colbar[comps+1]
> plot(g, vertex.size = 5, layout=layout_nicely)
```



Cut vertices and bridges Connectedness is an important property and it is useful to know what can "destroy" this property. For that, we seek vertices or edges whose deletion results in the graph becoming disconnected.

Definition 6.64 (Vertex deletion)**.** *If v is a vertex of G, the graph $G - v$ is the graph formed from G by removing v and all edges incident with v.*

Definition 6.65 (Cut-vertices)**.** *Let G be a connected graph. If $G - v$ is disconnected for $v \in V(G)$, then v is a **cut vertex** (or **articulation point**) of G.*

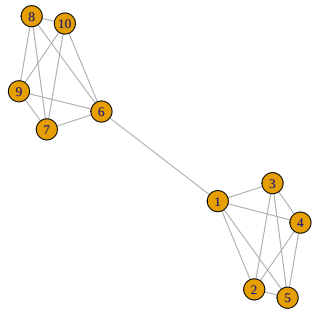
Definition 6.66 (Edge deletion)**.** *If e is an edge of G, the graph $G - e$ is the graph formed from G by removing e from G.*

Definition 6.67 (Bridge)**.** *An edge e in a connected graph G is called a **bridge** if $G - e$ is disconnected.*

Theorem 6.68. *Let G be a connected graph. An edge e of G is a bridge of G if and only if e does not lie on any cycle of G.*

In `igraph`, cut vertices are found using `articulation_points`, while bridges are found using `bridges`. Following an example from the `igraph` manual, we make a connected graph by making two full graphs of order 5 and joining them by selecting a vertex in each and making an edge between them.

```
> g <- disjoint_union( make_full_graph(5), make_full_graph(5) )
> clu <- components(g)$membership
> g <- add_edges(g, c(match(1, clu), match(2, clu)))
> plot(g)
```



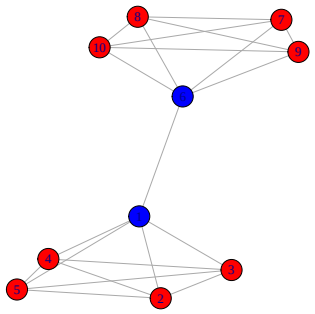
We then seek the cut vertices.

```
> articulation_points(g)
```

```
+ 2/10 vertices, from c540a5f:
[1] 6 1
```

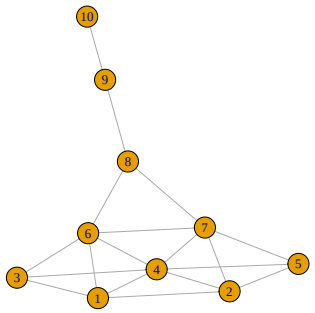
We can actually plot those vertices in a different colour.

```
> v_ap = articulation_points(g)
> V(g)$color <- "red"
> V(g)[v_ap]$color <- "blue"
> plot(g, layout=layout_nicely)
```



To illustrate bridges, let us make a special graph, the so-called Krackhardt kite graph.

```
> g <- make_graph("krackhardt_kite")
> plot(g)
```



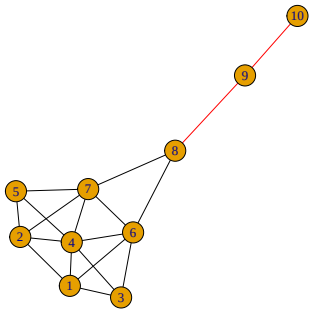
We then seek bridges,

```
> bridges(g)
```

```
+ 2/18 edges from 52f2699:
[1] 9--10 8-- 9
```

As we did in the previous example, let us highlight the bridges.

```
> e_b = bridges(g)
> E(g)$color = "black"
> E(g)[e_b]$color = "red"
> plot(g)
```



6.3.12 Planar graphs

Definition 6.69 (Planar graph)**.** *A graph is* ***planar*** *if it can be drawn in the plane with no crossing edges. Otherwise it is* ***nonplanar****.*

A planar graph is a graph that *can* be drawn in the plane. If a planar graph is indeed drawn in the plane, it is a plane graph.

Definition 6.70 (Plane graph)**.** *A plane graph is a graph that is drawn in the plane with no crossing edges. If G is a plane graph, then*

- *the connected parts of the plane are called* ***regions****;*
- *vertices and edges that are incident with a region R make up a* ***boundary*** *of R.*

Theorem 6.71 (Euler's formula)**.** *Let G be a connected plane graph with p vertices, q edges, and r regions, then*

$$p - q + r = 2.$$

Corollary 6.72. *Let G be a plane graph with p vertices, q edges, r regions, and k connected components, then*

$$p - q + r = k + 1.$$

Theorem 6.73. *Let G be a connected planar graph with p vertices and q edges, where $p \geq 3$, then*

$$q \leq 3p - 6.$$

(a maximal connected planar graph with p vertices has $q = 3p - 6$ edges.)

Corollary 6.74. *If G is a planar graph, then $\delta(G) \leq 5$, where $\delta(G)$ is the minimal degree of G. (every planar graph contains a vertex of degree less than 6)*

Definition 6.75 (Subdivision of G)**.** *Given a graph G, a subdivision of G is a graph that can be obtained by inserting any number of vertices of degree 2 along a edges of G.*

Theorem 6.76 (Kuratowski Theorem)**.** *A graph G is planar if and only if it contains no subgraph isomorphic to K_5 or $K_{3,3}$ or any subdivision of K_5 or $K_{3,3}$.*

Note: If a graph G is nonplanar and G is a subgraph of G', then G' is also nonplanar.

Definition 6.77 (Colouring of a graph G)**.**

- *assignment elements (colours) of some sets to vertices of G.*
- *one colour to each vertex so that adjacent vertices are assigned to different colours.*

(A colouring of a graph G is an assignment of colours to the vertices of G such that adjacent vertices have different colours.)

Definition 6.78 ($n-$colouring of G)**.** *A $n-$colouring is a colouring of G using n colours.*

Definition 6.79 ($n-$colourable)**.** *G is $n-$colourable if there exists a colouring of G using n colors.*

Definition 6.80 (Chromatic number)**.** *The **chromatic number** $\chi(G)$ of a graph G is the minimal value n for which an $n-$colouring of G exists.*

Theorem 6.81 (Some properties)**.** *Let $G = (V, E)$ be a graph, $\chi(G)$ its chromatic number.*

- *$\chi(G) = 1$ if and only if G has no edges.*
- *If $G = K_{n,m}$, then $\chi(G) = 2$.*
- *If $G = K_n$, then $\chi(G) = n$.*
- *For any graph G,*

$$\chi(G) \leq 1 + \Delta(G),$$

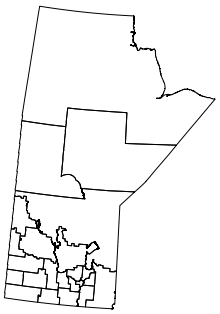
  where $\Delta(G)$ is the maximum degree of G.
- *If G is a planar graph, then $\chi(G) \leq 4$.*

Welch-Powell algorithm for coloring a graph G

1. Order the vertices of G in decreasing degree. Such an ordering may not be unique since some vertices may have the same degree.
2. Use one colour to paint the first vertex and to paint, in sequential order, each vertex on the list that is not adjacent to a vertex previously painted with this colour.
3. Start again at the top of the list and repeat the process painting previously unpainted vertices using a second colour.
4. Continue repeating with additional colours until all the vertices have been painted.

An actual real life colouring problem Let us consider the census divisions in Manitoba. Census divisions are the administrative regions that Statistics Canada uses when they are taking the census of the population of Canada every five years. Doing so will also illustrate how one can deal with maps and spatial information in R.

```
> library(dplyr)
> library(sf)
> census_divisions = st_read("DATA/lcd_000b21a_e/lcd_000b21a_e.shp") %>%
+   filter(PRUID == 46)
> plot(census_divisions$geometry)
```

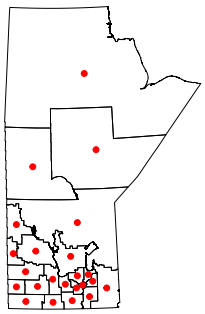


Right away, we notice an issue with this plot: the data is extracted from data for the entire country. This means that the projection that was used is perhaps a little extreme: Canada is a large country and most projections used to show the entire country distort local objects. The southern and northern borders of Manitoba are the 49th and 60th parallels, respectively. Both of these borders should be roughly horizontal. There are several ways to deal with this: rotate coordinates or change the projection used. The former is appealing in its simplicity, but the latter is more correct. And more annoying, since you will need to work out which projection to use. Looking around, we find a projection which should work.

```
> proj = "+proj=utm +zone=14 +datum=NAD83 +units=m +no_defs +type=crs"
> census_divisions = st_transform(census_divisions,
+                                 crs = proj)
```

We add the census division centroids, for good measure. Centroids are the centres of gravity of the regions, often used in the absence of more information about the largest cities in a region.

```
> centroids = st_centroid(census_divisions$geometry)
> plot(census_divisions$geometry)
> plot(centroids, add = TRUE, pch = 19, col = "red")
```



What is the minimal number of colours needed to colour this map in such a way that no two adjacent divisions have the same colour? To solve this problem, we need a mathematical representation. We proceed as follows.

- Vertices correspond to census divisions (in our representation earlier, we will choose the centroids of the regions).
- Two vertices are linked if the two census divisions are adjacent, i.e., they share a border.

GIS functions are very useful for this. If we have a collection of polygons (as the regions in the map are encoded), then `st_intersects` (in the `sf` library) finds polygons that intersect.

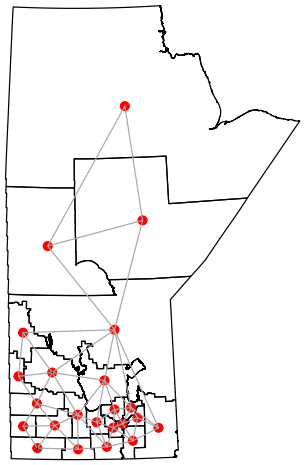
```
> intersects = st_intersects(census_divisions$geometry)
```

The result is a matrix, with each row showing what polygons the given polygon intersects with (including itself). Let us make a graph from this information. The matrix is a matrix of TRUE/FALSE values, let us transform it into an adjacency matrix.

```
> intersects.adj = mat.or.vec(nr = dim(intersects)[1],
+                             nc = dim(intersects)[2])
> intersects.adj[which(as.matrix(intersects) == TRUE)] = 1
> intersects.adj[which(as.matrix(intersects) == FALSE)] = 0
```

Now let us create the graph. To plot it, we need to set the position of the vertices, otherwise `igraph` will just lay them out randomly.

```
> g = graph_from_adjacency_matrix(intersects.adj, mode = "undirected")
> V(g)$x = unlist(centroids)[seq(1,length(unlist(centroids)), 2)]
> V(g)$y = unlist(centroids)[seq(2,length(unlist(centroids)), 2)]
> plot(census_divisions$geometry)
> plot(centroids, add = TRUE, pch = 19, col = "red")
> plot(g, add = TRUE, rescale = FALSE, vertex.label = NA)
```



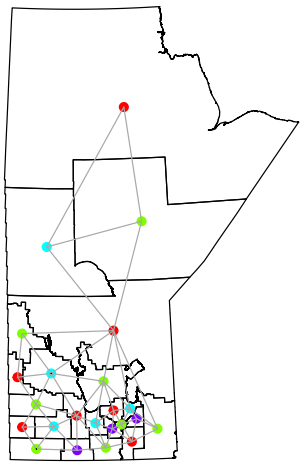
What is the chromatic number of the graph associated to the map?

We colourise the graph:

```
> v_c = greedy_vertex_coloring(g)
```

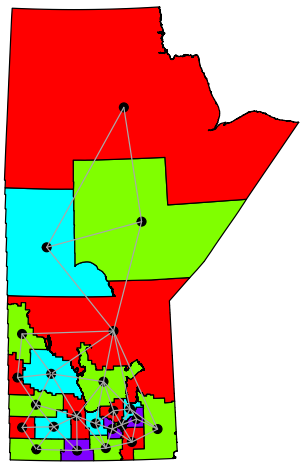
The result is a vector with each vertex assigned an integer value, i.e., a colour. Let us colour the vertices in the graph explicitly.

```
> colmap = rainbow(max(v_c))
> V(g)$col = colmap[v_c]
> plot(census_divisions$geometry)
> plot(centroids, add = TRUE, pch = 19, col = colmap[v_c])
> plot(g, add = TRUE, rescale = FALSE, vertex.label = NA)
```



Et voilà! Actually, just to make it clearer, let us colour the actual regions and revert to not colouring the vertices themselves.

```
> plot(census_divisions$geometry, col = colmap[v_c])
> plot(centroids, add = TRUE, pch = 19, col = "black")
> plot(g, add = TRUE, rescale = FALSE, vertex.label = NA)
```



Now, those of you who know the geography of Manitoba will have realised that there is an obvious problem here: this map is missing the two big lakes in the Province, Lake Winnipeg and Lake Manitoba. This could easily be addressed: get the shape files for lakes in the Province, add them to this map and impose that the vertices corresponding to lakes should be of the same colour. I will leave this as an exercise.

6.4 Directed graphs

6.4.1 Directed graph

Definition 6.82 (Digraph)**.** *A directed graph (or **digraph**) is a pair $G = (V, A)$ of sets such that*

- *V is a set of points: $V = \{v_1, v_2, v_3, .., v_p\}$*
- *A is a set of ordered pairs of V: $A = \{(v_i, v_j), (v_i, v_k), \ldots, (v_n, v_p)\}$, also noted $A = \{v_iv_j, v_iv_k, \ldots, v_nv_p\}$.*

Definition 6.83 (Vertex)**.** *The elements of V are the vertices of the digraph G. V is the vertex set of the digraph G, also noted $V(G)$.*

Definition 6.84 (Arc)**.** *The elements of A are the **arcs** (directed edges) of the digraph G. A is the arc set of the digraph G, also noted $A(G)$.*

Digraph and binary relation A digraph D can be defined in term of a vertex set V and an irreflexive relation R over V.

The defining relation R of the digraph G need not be symmetric.

Directed network

Definition 6.85 (Directed network)**.** *A directed network is a digraph together with a function f,*

$$f : A \to \mathbb{R},$$

which maps the arc set A into the set of real number. The value of the arc $uv \in A$ is $f(uv)$.

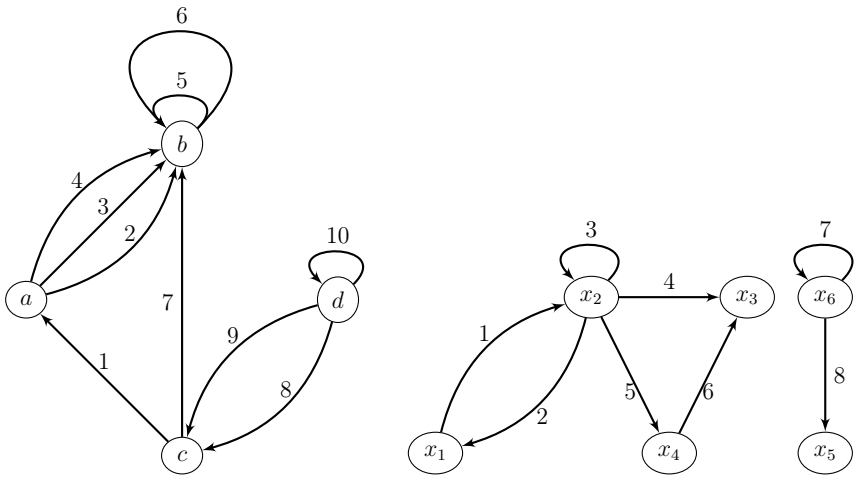
Loops & Multiple arcs

Definition 6.86 (Loop)**.** *A **loop** is an arc with both the same ends;* e.g. *(u, u) is a loop.*

Definition 6.87 (Multiple arcs)**.** ***Multiple arcs** (or multi-arcs) are two or more arcs connecting the same two vertices.*

Definition 6.88 (Multidigraph)**.** *A **multidigraph** is a digraph which allows repetition of arcs or loops.*

Definition 6.89 (Digraph)**.** *In a digraph, no more than one arc can join any pair of vertices.*



Examples

Let $G = (V, A)$ be a digraph.

Definition 6.90 (Arc endpoints)**.** *For an arc $u = (x, y)$, vertex x is the **initial endpoint**, and vertex y is the **terminal endpoint***

Definition 6.91 (Predecessor - Successor)**.** *If $(u, v) \in A(G)$ is an arc of G, then*

- *u is a **predecessor** of v,*
- *v is a **successor** of u.*

Definition 6.92 (Neighbours of a vertex)**.** *Let $x \in V$ be a vertex. The **neighbours** of x is the set $\Gamma(x) = \Gamma_G^+(x) \cup \Gamma_G^-(x)$, where $\Gamma_G^+(x)$ and $\Gamma_G^-(x)$ are, respectively, the set of successors and predecessors of v.*

Definition 6.93 (Directed away - Directed towards)**.** *If $a = (u, v) \in A(G)$ is an arc of G, then*

- *the arc a is said to be directed away from u,*
- *the arc a is said to be directed towards v.*

Definition 6.94 (Source - Sink)**.**

- *Any vertex which has no arcs directed towards it is a **source**.*
- *Any vertex which has no arcs directed away from it is a **sink**.*

Definition 6.95 (Adjacent arcs)**.** *Two arcs are **adjacent** if they have at least one endpoint in common.*

Definition 6.96 (Arc incident out of $A \subset X$)**.** *If the initial endpoint of an arc u belongs to A, and if the terminal endpoint of arc u does not belong to A, then u is said to be incident out of A, and we write $u \in \omega^+(A)$. Similarly, we define an arc incident into A, and the set $\omega^-(A)$. Finally, the set of arcs incident to A is denoted*

$$\omega(A) = \omega^+(A) \cup \omega^-(A).$$

Definition 6.97 (Symmetric graph)**.** *If $m_G^+(x, y) = m_G^-(x, y)$ for all $x, y \in X$, the graph G is **symmetric**. A 1-graph $G = (X, U)$ is symmetric if, and only if,*

$$(x, y) \in U \implies (y, x) \in U$$

Definition 6.98 (Anti-symmetric graph)**.** *If for each pair* $(x, y) \in X \times X$*,*

$$m_G^+(x, y) + m_G^-(x, y) \leq 1$$

then the graph G *is* ***anti-symmetric****. A 1-graph* $G = (X, U)$ *is anti-symmetric if, and only if,*

$$(x, y) \in U \implies (y, x) \notin U$$

An anti-symmetric 1-graph without its direction is a simple graph

Definition 6.99 (Subgraph of G generated by $A \subset X$)**.** *The* ***subgraph*** *of* G *generated by* A *is the graph with* A *as its vertex set and with all the arcs in* G *that have both their endpoints in* A*. If* $G = (X, \Gamma)$ *is a 1-graph, then the subgraph generated by* A *is the 1-graph* $G_A = (A, \Gamma_A)$ *where*

$$\Gamma_A(x) = \Gamma(x) \cap A \qquad (x \in A)$$

Definition 6.100 (Partial graph of G generated by $V \subset U$)**.** *The graph* (X, V) *whose vertex set is* X *and whose arc set is* V*. In other words, it is graph* G *without the arcs* $U - V$

Definition 6.101 (Partial subgraph of G)**.** *A partial subgraph of* G *is the subgraph of a partial graph of* G

6.4.2 Degrees in digraphs

Let v be a vertex of a digraph $G = (V, A)$.

Definition 6.102 (Outdegree of a vertex)**.** *The number of arcs directed away from a vertex* v*, in a digraph is called the* ***outdegree*** *of* v *and is written* $d^+(v)$ *or* $outdeg(v)$*.*

Definition 6.103 (Indegree of a vertex)**.** *The number of arcs directed towards a vertex* v*, in a digraph is called the* ***indegree*** *of* v *and is written* $d^-(v)$ *or* $indeg(v)$*.*

Definition 6.104 (Degree)**.** *For any vertex* v *in a digraph, the* ***degree*** *of* v *is defined as* $d(v) = d^+(v) + d^-(v)$*.*

Theorem 6.105. *For any (di)graph, the sum of the degrees of the vertices equals twice the number of edges (arcs).*

Corollary 6.106. *In any (di)graph, the sum of the degrees of the vertices is a nonnegative even integer.*

Theorem 6.107. *If* G *is a digraph with a vertex set* $V(G) = \{v_1, \ldots, v_p\}$ *and having* q *arcs then*

$$\sum_{i=1}^{p} d^+(v_i) = \sum_{i=1}^{p} d^-(v_i) = q.$$

Definition 6.108 (Regular digraph)**.** *A digraph* G *is* $r-$*regular if* $indeg(v) = outdeg(v) = r$ *for each vertex* v *of* G*.*

6.4.3 Walks, paths, etc.

Let $G = (V, A)$ be a digraph.

Definition 6.109 (Directed walk)**.** *A **directed walk** in a digraph G is a non-empty alternating sequence $v_0 a_0 v_1 a_1 v_2 \dots a_{k-1} v_k$ of vertices and arcs in G such that $a_i = (v_i, v_{i+1})$ for all $i < k$. This walk begins with v_0 and ends with v_k.*

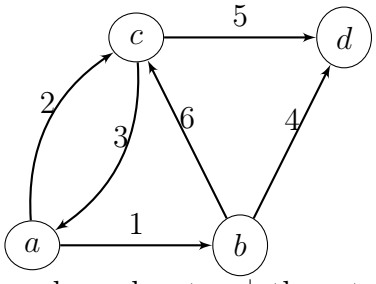
Definition 6.110 (Length of a directed walk)**.** *The length of a directed walk is equal to the number of arcs in the directed walk.*

Definition 6.111 (Closed walk)**.** *If $v_0 = v_k$, the walk is closed.*

Definition 6.112 (Directed trail)**.** *A directed walk in G in which all arcs are distinct is a **directed trail** in G.*

Definition 6.113 (Directed path)**.** *A directed walk in G in which all vertices are distinct is a **directed path** in G.*

Definition 6.114 (Directed cycle)**.** *A closed walk is a **directed cycle** if it contains at least three vertices and all its vertices are distinct except for $v_0 = v_k$.*



Cycles:

- $\boldsymbol{\mu}^1 = (1, 6, 2) = [abca]$
- $\boldsymbol{\mu}^2 = (1, 6, 3) = [abca]$
- $\boldsymbol{\mu}^3 = (2, 3) = [aca]$
- $\boldsymbol{\mu}^4 = (1, 4, 5, 2) = [abdca]$
- $\boldsymbol{\mu}^5 = (6, 5, 4) = [acdb]$
- $\boldsymbol{\mu}^6 = (1, 4, 5, 3) = [abdca]$

Given a cycle μ, denote μ^+ the set of all arcs in μ that are in the direction that the cycle is traversed and μ^- the set of all the other arcs in μ

Number the arcs in G as $1, 2, \dots, m$, then the cycle μ is the vector

$$\mathbf{m}\mu = (\mu_1, \dots, \mu_m)$$

where

$$\mu_i = \begin{cases} 0 & \text{if } i \notin \mu^+ \cup \mu^- \\ +1 & \text{if } i \in \mu^+ \\ -1 & \text{if } i \in \mu^- \end{cases}$$

Cocycles Let $A \subset X$ be nonempty and denote $\omega^+(A)$ the set of arcs that have only their initial endpoint in A and $\omega^-(A)$ the set of arcs that have only their terminal endpoint in A. Let

$$\omega(A) = \omega^+(A) \cup \omega^-(A)$$

A **cocycle** is a nonempty set of arcs of the form $\omega(A)$, partitioned into two sets $\omega^+(A)$ and $\omega^-(A)$

An **elementary cocycle** is the set of arcs joining two connected subgraphs A_1 and A_2 s.t.

- $A_1, A_2 \neq \emptyset$
- $A_1 \cap A_2 = \emptyset$
- $A_1 \cup A_2 = C$, with C a connected component of the graph

A colouring lemma

Lemma 6.115 (Arc colouring Lemma)**.** *Consider G with arcs $1, \ldots, m$. Colour arc 1 black and arbitrarily colour the remaining arcs red, black or green. Then exactly one of the following holds true:*

1. *there is an elementary cycle containing arc 1 and only red and black arcs with the property that all black arcs in the cycle have the same direction*
2. *there is an elementary cocycle containing arc 1 and only green and black arcs, with the property that all black arcs in the cocycle have the same direction*

Independent cycles and cycle bases Consider cycles $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k$. The cycles are **independent** if

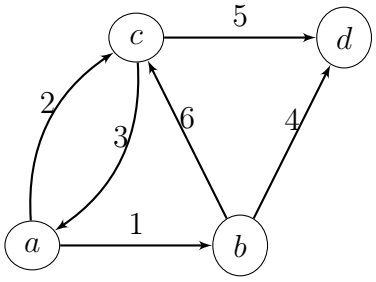
$$c_1\boldsymbol{\mu}^1 + c_2\boldsymbol{\mu}^2 + \cdots + c_k\boldsymbol{\mu}^k = \mathbf{0}$$
$$\iff c_1 = c_2 = \cdots = c_k = 0$$

A **cycle basis** is an independent set $\{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k\}$ of cycles such that any cycle $\boldsymbol{\mu}$ can be written as

$$\boldsymbol{\mu} = c_1\boldsymbol{\mu}^1 + c_2\boldsymbol{\mu}^2 + \cdots + c_k\boldsymbol{\mu}^k$$

for $c_1, \ldots, c_k \in \mathbb{R}$

The constant k is the **cyclomatic number** of G, denoted $\nu(G)$



Elementary cycles:

- $\boldsymbol{\mu}^1 = (1, 6, 2) = [abca]$
- $\boldsymbol{\mu}^2 = (1, 6, 3) = [abca]$
- $\boldsymbol{\mu}^3 = (2, 3) = [aca]$
- $\boldsymbol{\mu}^4 = (1, 4, 5, 2) = [abdca]$
- $\boldsymbol{\mu}^5 = (6, 5, 4) = [acdb]$
- $\boldsymbol{\mu}^6 = (1, 4, 5, 3) = [abdca]$

We have $\boldsymbol{\mu}^1 - \boldsymbol{\mu}^2 + \boldsymbol{\mu}^3 = \mathbf{0}$

Theorem 6.116. *Let G be a graph with n vertices, m arcs and p connected components. Then the cyclomatic number of G is*

$$\nu(G) = m - n + p.$$

6.4.4 Connectivity in digraphs

Definition 6.117 (Underlying graph)**.** *Given a digraph, the undirected graph with each arc replaced by an edge is called the **underlying graph**.*

Definition 6.118 (Weakly connected digraph)**.** *If the underlying graph is a connected graph, then the digraph is **weakly connected**.*

Definition 6.119 (Strongly connected digraph)**.** *A digraph G is **strongly connected** if for every two distinct vertices u and v of G, there exists a directed path from u to v.*

Definition 6.120 (Disconnected digraph)**.** *A digraph is said to be **disconnected** if it is not weakly connected.*

Strong connectedness is an equivalence relation Denote $x \equiv y$ the relation "$x = y$, or $x \neq y$ and there exists a directed path in G from x to y". $\equiv$ is an equivalence relation since

1. $x \equiv y$ [reflexivity]
2. $x \equiv y \implies y \equiv x$ [symmetry]
3. $x \equiv y, y \equiv z \implies x \equiv z$ [transitivity]

Definition 6.121 (Connected component of a graph)**.** *Sets of the form*

$$A(x_0) = \{x : x \in X, x \equiv x_0\}$$

*are equivalence classes. They partition X into strongly connected sub-digraphs of G called **strongly connected components** (or **strong components**) of G.*

A strong component in G is a maximal strongly connected subdigraph of G.

Let $G = (V, A)$ be a digraph.

Theorem 6.122. *Properties*

- *If a digraph is strongly connected, it has only one strongly connected component.*
- *The strongly connected components partition the vertices in the digraph, with every vertex in exactly one strongly connected component.*

Algorithm for determining strongly connected components in $G = (V, A)$

- Determine the strongly connected component $C(v)$ containing the vertex v; if $V - C(v)$ is non-empty, re-do the same operation on the sub-digraph $G' = (V - C(v), A')$.
- To determine $C(v)$, the strongly connected component containing v: let v be a vertex of a digraph , which is not already in any strongly connected component.
  1. Mark the vertex v with $\pm$
  2. Mark with $+$ all successors (not already marked with $+$) of a vertex marked with $+$
  3. Mark with $-$ all predecessors (not already marked with $-$) of a vertex marked with $-$
  4. Repeat until no more possible marking with $+$ or $-$

  All vertices marked with $\pm$ belong to the same strongly connected component $C(v)$ containing the vertex v.

Definition 6.123 (Condensation of a digraph)**.** *The condensation G^* of a digraph G is a digraph having as vertices the strongly connected components (SCC) of G and such that there exists an arc in G^* from a SCC C_i to another SCC C_j if there is an arc in G from some vertex of S_i to a vertex of S_j.*

Definition 6.124 (Articulation set)**.** *For a connected graph, a set A of vertices is called an* ***articulation set*** *(or a* ***cutset****) if the subgraph of G generated by $X - A$ is not connected.*

Definition 6.125 (Stable set)**.** *A set S of vertices is called a stable set if no arc joins two distinct vertices in S.*

6.4.5 Orientable graphs

In much the same way as a directed graph can be considered as an undirected one by considering the *underlying graph* (Definition 6.117), one can transform an undirected graph into a directed one.

Definition 6.126 (Orienting a graph)**.** *Given a connected* graph*, we describe the act of assigning a direction to each edge as* ***orienting the graph****.*

Definition 6.127 (Strong orientation)**.** *If orienting the graph gives a digraph that is strongly connected, the orientation is a* ***strong orientation****.*

Definition 6.128 (Orientable graph)**.** *A connected graph $G = (V, E)$ is* ***orientable*** *if it is possible to assign a direction to each edge of G to produce a strongly connected digraph $G = (V, A)$. (If there exists a strong orientation of a connected graph, then the graph is orientable.)*

Theorem 6.129. *A connected graph G is orientable (has a strong orientation) if and only if G contains no bridges; that is every edge is contained in a cycle.*

6.5 Trees

Definition 6.130 (Forest, trees and branches)**.**

- *A connected graph with no cycle is a* ***tree****.*
- *A tree is a connected acyclic graph, its edges are called* ***branches****.*
- *A graph (connected or not) without any cycle is a* ***forest****. Each component is a tree. (A forest is a graph whose connected components are trees)*

Theorem 6.131 (Properties)**.**

- *Every edge of a tree is a bridge (the deletion of any edge of a tree diconnects it)*
- *Given two vertices u and v of a tree, there is an unique path linking u to v.*
- *A tree with p vertices and q edges satisfies $q = p - 1$. Thus, a tree is minimally connected.*

Definition 6.132 (Spanning tree)**.** *A* ***spanning tree*** *of a connected graph G is a subgraph of G that contains all the vertices of G and is a tree.*

A graph may have many spanning trees.

Definition 6.133 (Value of a spanning tree)**.** *The **value of a spanning tree** T of order p is*

$$\sum_{i=1}^{p-1} f(e_i)$$

where f is the function that maps the edge set into the set of real number.

Definition 6.134 (Minimal spanning tree)**.** *Let G be an undirected network, and let T be a **minimal spanning tree** of G. Then T is a spanning tree whose the value is minimum.*

Algorithm to find a minimal spanning tree Let $G = (V, E)$ be an undirected network, and let T be a minimal spanning tree.

1. sort the edges of G in increasing order by value
2. $T = (V(G), \emptyset)$
3. for each edge e in sorted order if the endpoints of e are disconnected in T add e to T

Minimal connector problem

- Model: a graph G such that edges represent all possible connections, and each edge has a positive value which represents its cost; an undirected network G
- Solution: a minimal spanning tree T of G
  - a spanning tree of G is a subgraph of G that contains all the vertices of G and is a tree.
  - the cost of the spanning tree is the sum of values of the edges of T
  - a spanning tree such that no other spanning tree has a smaller cost is a minimmal spanning tree.

Theorem 6.135 (Characterisation of trees)**.** *$H = (X, U)$ a graph of order $|X| = n > 2$. The following are equivalent and all characterise a tree :*

1. *H connected and has no cycles*
2. *H has $n - 1$ arcs and no cycles*
3. *H connected and has exactly $n - 1$ arcs*
4. *H has no cycles, and if an arc is added to H, exactly one cycle is created*
5. *H connected, and if any arc is removed, the remaining graph is not connected*
6. *Every pair of vertices of H is connected by one and only one chain*

Proof of Theorem 6.135. We make abundant use of Theorem 6.116
(**1** $\implies$ **2**) Let p be the number of connected components, m the number of arcs and $\nu(H)$ the cyclomatic number. Since H connected, $p = 1$. Since H has no cycles, $\nu(H) = m - n + p = 0 \implies m = n - p = n - 1$
(**2** $\implies$ **3**) Assume H has no cycles ($\nu(H) = 0$) and has $n - 1$ arcs ($m = n - 1$). Then, since

$$\nu(H) = m - n + p,$$

$p = \nu(H) - m + n = 0 - (n-1) - n = 1$, i.e., H is connected.
(**3** $\implies$ **4**) Assume H connected ($p = 1$) and contains exactly $n-1$ arcs ($m = n-1$). Then

$$\nu(H) = m - n + p = (n-1) - n + 1 = 0$$

and H has no cycles. Now add an arc, i.e., suppose $m = n$. Then $\nu(H) = m - n + p = n - n + 1 = 1$ and there is one cycle in the new graph.
(**4** $\implies$ **5**) Assume H has no cycles ($\nu(H) = 0$) and that addition of an arc to H creates exactly one cycle. Suppose H not connected. Then there are two vertices, say a and b, that are not connected and adding the arc (a, b) does not create a cycle, a contradiction with "addition of an arc to H creates exactly one cycle". $\implies$ $p = 1$. Since $\nu(H) = 0$, this implies that $m = n - 1$. Now suppose we remove an arc. We obtain graph H' with

$$m' = n' - 2 \quad \text{and} \quad \nu(H') = 0.$$

So

$$p' = \nu(H') - m' + n' = 2,$$

which implies that H' is not connected.
(**5** $\implies$ **6**) Assume H connected and if any arc is removed, the remaining graph is not connected.

Any vertices $a, b \in X$ are connected by a chain (since H connected). That chain is unique: suppose there is a second chain connecting a to b; then removing an arc from that chain does not disconnect the graph, since there is still the original chain connecting a and b.
(**6** $\implies$ **1**) Assume every pair of vertices of H is connected by one and only one chain. Now assume H has a cycle. Then at least one pair of vertices would be connected by two distinct chains, a contradiction. □

Definition 6.136 (Pendant vertex)**.** *A vertex is* ***pendant*** *if it is adjacent to exactly one other vertex.*

Theorem 6.137. *A tree of order $n \geq 2$ has at least two pendant vertices.*

Proof. Suppose H is a tree of order $n \geq 2$ with 0 or 1 pendant vertices. Consider a traveller traversing the graph edges, starting from a pendant vertex (if there is one) or anywhere if there is no pendant vertex. If the traveller does not allow himself to use same edge twice, he cannot go to the same vertex twice, since H has no cycle. If he arrives at vertex x, he can depart x using a new edge (x is not a pendant vertex as there are 0 or 1 pendant vertex and if there is 1, that's where he started). So the trip continues without end, which is impossible, as H finite. □

Theorem 6.138. *A graph $G = (X, U)$ has a partial graph that is a tree $\iff$ G connected*

Recall that a partial graph is a graph generated by a subset of the arcs (Definition 6.100 slide 90)

Proof. Suppose that G is not connected. Then no partial graph of G is connected $\implies$ there is no partial graph of G that is a tree. [We want to show $P \iff Q$, we start with $\wedge Q \implies \wedge P$ (which $\iff P \implies Q$)] Suppose that G connected. Look for an arc whose removal does not disconnect G

- if there is none, G is a tree by Theorem 6.135(5)
- if there is, remove it and look for another one, etc. When no more arcs can be removed, the remaining graph is a tree with vertex set X. □

The procedure in the proof of Theorem 6.138 gives a **spanning tree** It is also possible to build a spanning tree as follows:

- Consider any arc u_0.
- Find arc u_1 that does not form a cycle with u_0.
- Find arc u_2 that does not form a cycle with $\{u_0, u_1\}$.
- Continue.
- When you cannot continue anymore, you have a spanning tree.

Theorem 6.139. *G connected graph with ≥ 1 arc. The following are equivalent.*

1. *G is strongly connected.*
2. *Every arc lies on a circuit.*
3. *G contains no cocircuits.*

Proof. (**1** $\implies$ **2**) (x, y) an arc of G; there is a path from y to x (G strongly connected), so arc (x, y) is contained in a circuit of G.

(**2** $\implies$ **3**) Suppose G has a cocircuit containing arc (x, y); then G cannot have a circuit containing this arc by the Arc Colouring Lemma (Lemma 6.115 with all arcs coloured black. This contradicts (2).

(**3** $\implies$ **1**) Assume G is a connected graph without cocircuits, but that G is not strongly connected. Since G is not strongly connected, it has more than one strongly connected component. Since G is connected, there exist two distinct strongly connected components that are joined by an arc (a, b). The arc (a, b) is not contained in any circuit; indeed, otherwise a and b would be in the same strongly connected component. By Lemma 6.115, arc (a, b) is contained in some cocircuit. This contradicts (3). □

Theorem 6.140. *G graph with ≥ 1 arc. The following are equivalent.*

1. *G is a graph without circuits.*
2. *Each arc is contained in a cocircuit.*

Theorem 6.141. *If G is a strongly connected graph of order n, then G has a cycle basis of $\nu(G)$ circuits.*

Definition 6.142 (Node, anti-node, branch)**.** *$G = (X, U)$ strongly connected without loops and > 1 vertex. For each $x \in X$, there is a path from it and a path to it so x has at least 2 incident arcs. Specifically,*

- *$x \in X$ with > 2 incident arcs is a* ***node***
- *$x \in X$ with 2 incident arcs is an* ***anti-node***

A path whose only nodes are its endpoints is a ***branch***

Definition 6.143 (Minimally connected graph)**.** *G is* ***minimally connected*** *if it is strongly connected and removal of any arc destroys strong-connectedness*

A minimally connected graph is 1-graph without loops

Definition 6.144 (Contraction)**.** *$G = (X, U)$. The* ***contraction*** *of the set $A \subset X$ of vertices consists in replacing A by a single vertex a and replacing each arc into (resp. out of) A by an arc with same index into (resp. out of) a*

Theorem 6.145. *G minimally connected, $A \subset X$ generating a strongly connected subgraph of G. Then the contraction of A gives a minimally connected graph*

Proof. First, show that contraction of A yields a 1-graph. If this were not the case, there would exist $x \notin A$ and $a, a' \in A$ s.t. $(x, a), (x, a') \in U$ (or, with $(a, x), (a', x) \in U$ but this would not change the proof). If one of these arcs is removed, the graph remains strongly connected. Thus, G is not minimally connected, a contradiction

Now show that the contraction of A yields a graph G' that is minimally connected. Clearly, G' strongly connected. If an arc u is removed, the remaining graph is not strongly connected, since the graph $(X, U - \{u\})$ not strongly connected. □

Theorem 6.146. *G a minimally connected graph, G' be the minimally connected graph obtained by the contraction of an elementary circuit of G. Then*

$$\nu(G) = \nu(G') + 1$$

Theorem 6.147. *G minimally connected of order $n \geq 2 \implies G$ has ≥ 2 anti-nodes*

Theorem 6.148. *$G = (X, U)$. Then the graph C' obtained by contracting each strongly connected component of G contains no circuits*

Definition 6.149 (Root)**.** *Vertex $a \in X$ in $G = (X, U)$ is a* ***root*** *if all vertices of G can be reached by paths* starting *from a*

Not all graphs have roots.

Definition 6.150 (Quasi-strong connectedness)**.** *G is* ***quasi-strongly connected*** *if $\forall x, y \in X$, exists $z \in X$ (denoted $z(x, y)$ to emphasize dependence on x, y) from which there is a path to x and a path to y*

Strongly connected $\implies$ quasi-strongly connected (take $z(x, y) = x$); converse not true

Quasi-strongly connected $\implies$ connected.

Definition 6.151 (Arborescence)**.** *An* ***arborescence*** *is a tree that has a root.*

Lemma 6.152. *$G = (X, U)$ has a root $\iff$ G quasi-strongly connected.*

Theorem 6.153. *H graph of order $n > 1$. TFAE (and all characterise an arborescence)*

1. *H quasi-strongly connected without cycles*
2. *H quasi-strongly connected with $n-1$ arcs*
3. *H tree having a root a*
4. *$\exists a \in X$ s.t. all other vertices are connected with a by 1 and only 1 path from a*
5. *H quasi-strongly connected and loses quasi-strong connectedness if any arc is removed*
6. *H quasi-strongly connected and $\exists a \in X$ s.t.*
$$\begin{aligned} d_H^-(a) &= 0 \\ d_H^-(x) &= 1 \qquad \forall x \neq a \end{aligned}$$
7. *H has no cycles and $\exists a \in X$ s.t.*
$$\begin{aligned} d_H^-(a) &= 0 \\ d_H^-(x) &= 1 \qquad \forall x \neq a \end{aligned}$$

Theorem 6.154. *G has a partial graph that is an arborescence $\iff$ G quasi-strongly connected.*

Theorem 6.155. *$G = (X, E)$ simple connected graph and $x_1 \in X$. It is possible to direct all edges of E so that the resulting graph $G_0 = (X, U)$ has a spanning tree H s.t.*

1. *H is an arborescence with root x_1*
2. *The cycles associated with H are circuits*
3. *The only elementary circuits of G_0 are the cycles associated with H*

Counting trees

Proposition 6.156. *X a set with n distinct objects, $n_1, \ldots, n_p$ nonnegative integers s.t. $n_1 + \cdots + n_p = n$. The number of ways to place the n objects into p boxes $X_1, \ldots, X_p$ containing $n_1, \ldots, n_p$ objects respectively is*
$$\binom{n}{n_1, \ldots, n_p} = \frac{n!}{n_1! \cdots n_p!}$$

Proposition 6.157 (Multinomial formula)**.** *Let $a_1, \ldots, a_p \in \mathbb{R}$ be p real numbers, then*
$$(a_1 + \cdots + a_p)^n = \sum_{n_1, \ldots, n_p \geq 0} \binom{n}{n_1, \ldots, n_p} (a_1)^{n_1} \cdots (a_p)^{n_p}$$

Theorem 6.158. *Denote $T(n; d_1, \ldots, d_n)$ the number of distinct trees H with vertices $x_1, \ldots, x_n$ and with degrees $d_H(x_1) = d_1, \ldots, d_H(x_n) = d_n$. Then*
$$T(n; d_1, \ldots, d_n) = \binom{n-2}{d_1 - 1, \ldots, d_n - 1}$$

Theorem 6.159. *The number of different trees with vertices $x_1, \ldots, x_n$ is n^{n-2}*

There is a whole industry of similar results (as well as for arborescences), but we will stop here. The main point is that we are talking about a large number of possibilities.

6.6 Matrices associated to a graph/digraph

There are multiple matrices associated to a graph/digraph. The branch of graph theory that studies the properties of matrices derived from graphs and uses of these matrices in determining graph properties is *spectral graph theory*. Graphs greatly simplify some problems in linear algebra and vice versa. We briefly explore some of these relationships here.

6.6.1 Adjacency matrices

Adjacency matrix (undirected case) Let $G = (V, E)$ be a graph of order p and size q, with vertices $v_1, v_2, \ldots, v_p$ and edges $e_1, e_2, \ldots, e_q$.

Definition 6.160 (Adjacency matrix)**.** *The adjacency matrix is*

$$M_A = M_A(G) = [m_{ij}]$$

is a $p \times p$ matrix in which

$$m_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Theorem 6.161 (Adjacency matrix and degree)**.** *The sum of the entries in row i of the adjacency matrix is the degree of v_i in the graph.*

Adjacency matrix of a multigraph

Definition 6.162 (Multiplicity of a pair)**.** *The **multiplicity** of a pair x, y is the number $m_G^+(x, y)$ of arcs with initial endpoint x and terminal endpoint y. Let*

$$m_G^-(x, y) = m_G^+(y, x)$$
$$m_G(x, y) = m_G^+(x, y) + m_G^-(x, y)$$

If $x \neq y$, then $m_G(x, y)$ is number of arcs with both x and y as endpoints. If $x = y$, then $m_G(x, y)$ equals twice the number of loops attached to vertex x. If $A, B \subset X$, $A \neq B$, let

$$m_G^+(A, B) = \{u : u \in U, u = (x, y), x \in A, y \in B\}$$
$$m_G(A, B) = m_G^+(A, B) + m_G^+(A, B)$$

Definition 6.163 (Matrix associated with G)**.** *If G has vertices $x_1, x_2, \ldots, x_n$, then the **matrix associated** with G is*

$$a_{ij} = m_G^+(x_i, x_j)$$

Definition 6.164 (Adjacency matrix)**.** *The matrix $a_{ij} + a_{ji}$ is the* ***adjacency matrix*** *associated with G*

Let $D = (V, A)$ be a digraph of order p with vertices denoted by $v_1, v_2, \ldots, v_p$.

Definition 6.165 (Adjacency matrix)**.** *The adjacency matrix $M = M(D) = [m_{ij}]$ is a $p \times p$ matrix in which*

$$m_{ij} = \begin{cases} 1 & \text{if arc } v_i v_j \in A \\ 0 & \text{otherwise} \end{cases}$$

Theorem 6.166 (Properties)**.**

- *M is not necessarily symmetric.*
- *The sum of any column of M is equal to the number of arcs directed towards v_j*
- *The sum of the entries in row i is equal to the number of arcs directed away from vertex v_i.*
- *The $(i, j)-$entry of M^n is equal to the number of walks of length n from vertex v_i to v_j.*

Incidence matrix Let $D = (V, A)$ be a digraph of order p, and size q, with vertices denoted by $v_1, v_2, \ldots, v_p$, and arcs denoted $a_1, a_2, \ldots, a_q$.

Definition 6.167. *Definition The incidence matrix $B = B(D) = [b_{ij}]$ is a $p \times q$ matrix in which*

$$b_{ij} = \begin{cases} 1 & \text{if arc } a_j \text{ is directed away from a vertex } v_i \\ -1 & \text{if arc } a_j \text{ is directed towards a vertex } v_i \\ 0 & \text{otherwise} \end{cases}$$

We have already seen adjacency matrices, let us recall the definition here

Definition 6.168 (Adjacency matrix)**.** *G a 1-graph, then the* ***adjacency matrix*** *$A = [a_{ij}]$ is defined as follows*

$$a_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \in U \\ 0 & \text{otherwise} \end{cases}$$

We often write $A(G)$ and, reciprocally, if A is an adjacency matrix, $G(A)$ the corresponding graph

G undirected $\implies$ $A(G)$ symmetric

$A(G)$ has nonzero diagonal entries if G is not simple

Adjacency matrix (multigraph case)

Definition 6.169 (Adjacency matrix of a multigraph)**.** *G an ℓ-graph, then the adjacency matrix $M_A = [m_{ij}]$ is defined as follows*

$$m_{ij} = \begin{cases} k & \text{if arc there are } k \text{ arcs } (i, j) \in U \\ 0 & \text{otherwise} \end{cases}$$

with $k \leq \ell$

G undirected $\implies$ $M_A(G)$ symmetric

$M_A(G)$ has nonzero diagonal entries if G is not simple.

Weighted adjacency matrices Sometimes, adjacency matrices (typically for 1-graphs) have real entries, usually positive

This means that the arcs/edges have been given a weight

Theorem 6.170 (Number of walks of length n)**.** *Let A be the adjacency matrix of a graph $G = (V(G), E(G))$, where $V(G) = \{v_1, v_2, \ldots, v_p\}$. Then the $(i,j)-$entry of A^n, $n \geq 1$, is the number of different walks linking v_i to v_j of length n in G.*

(two walks of the same length are equal if their edges occur in exactly the same order)

Example: let A be the adjacency matrix of a graph $G = (V(G), E(G))$.

- the $(i,i)-$entry of A^2 is equal to the degree of v_i.
- the $(i,i)-$entry of A^3 is equal to twice the number of C_3 containing v_i.

6.6.2 Other matrices associated to a graph/digraph

Let $G = (V, E)$ be a graph of order p and size q, with vertices $v_1, v_2, \ldots, v_p$, and edges $e_1, e_2, \ldots, e_q$.

Definition 6.171 (Incidence matrix)**.** *The incidence matrix is*

$$B = B(G) = [b_{ij}]$$

is that $p \times q$ matrix in which

$$b_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is incident with } e_j \\ 0 & \text{otherwise} \end{cases}$$

Theorem 6.172 (Incidence matrix and degrees)**.** *The sum of the entries in row i of the incidence matrix is the degree of v_i in the graph.*

Definition 6.173 (Incidence matrix – Undirected case)**.** *$G = (X, E)$ an undirected graph of order n with p edges. The **incidence** matrix of G is an $n \times p$ matrix with vertices as rows and edges as columns and where $B = [b_{ij}]$ satisfies*

$$b_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is incident to vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Definition 6.174 (Incidence matrix – Directed case)**.** *$G = (X, U)$ a directed graph of order n with p arcs. The **incidence** matrix of G is an $n \times p$ matrix with vertices as rows and edges as columns and where $B = [b_{ij}]$ satisfies*

$$b_{ij} = \begin{cases} 1 & \text{if arc } j \text{ "enters" vertex } i \\ -1 & \text{if arc } j \text{ "leaves" vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Definition 6.175 (Spectrum of a graph)**.** *The* ***spectrum of a graph*** $G = (V, X)$ *is the spectrum (set of eigenvalues) of its associated adjacency matrix* $A(G)$

This is regardless of the type of adjacency matrix or graph.

Definition 6.176 (Degree matrix)**.** *The* ***degree matrix*** $D = [d_{ij}]$ *for* G *is a* $n \times n$ *diagonal matrix defined as*

$$d_{ij} = \begin{cases} d_G(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

In an undirected graph, this means that each loop increases the degree of a vertex by two. In a directed graph, the term "degree" may refer either to indegree (the number of incoming edges at each vertex) or outdegree (the number of outgoing edges at each vertex).

Definition 6.177 (Laplacian matrix)**.** *Let* $G = (X, U)$ *be a simple graph of order* n*. The* ***Laplacian matrix*** *is*

$$L(G) = D(G) - A(G),$$

where $D(G)$ *is the degree matrix and* $A(G)$ *is the adjacency matrix.*

G simple graph $\implies$ $A(G)$ only contains 1 or 0 and its diagonal elements are all 0. For directed graphs, either the indegree or outdegree is used, depending on the application. The elements of L are given by,

$$\ell_{ij} = \begin{cases} d_G(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Distance matrix Let G be a graph of order p with vertices denoted by $v_1, v_2, \ldots, v_p$.

Definition 6.178 (Distance matrix)**.** *The distance matrix* $DIST = DIST(D) = [d_{ij}]$ *is a* $p \times p$ *matrix in which*

$$d_{ij} = dist(v_i, v_j).$$

Note $d_{ii} = 0$ *for* $i = 1, \cdots, p$.

Definition 6.179 (Properties)**.**

- *M is not necessarily symmetric.*
- *The sum of any column of M is equal to the number of arcs directed towards v_j*
- *The sum of the entries in row i is equal to the number of arcs directed away from vertex v_i.*
- *The $(i, j)-$entry of M^n is equal to the number of walks of length n from vertex v_i to v_j.*

6.6.3 Linking graphs and linear algebra

Counting paths To count paths between vertices x and y in a graph, we use the adjacency matrix

Theorem 6.180. *$G = (V, X)$ a graph and $A(G)$ its adjacency matrix. Denote $P = [p_{ij}]$ the matrix $P = A^k$. Then p_{ij} is the number of distinct paths of length k from i to j in G.*

In the case of a digraph, p_{ij} is the number of directed paths. In linear algebra, there is a corresponding notion, the *(ir)reducibility* of a matrix.

Definition 6.181 (Irreducible matrix)**.** *A matrix $A \in \mathcal{M}_n$ is **reducible** if $\exists P \in \mathcal{M}_n$, permutation matrix, such that $P^T AP$ can be written in block triangular form. If no such P exists, A is **irreducible**.*

The following result then provides a fundamental connection between graphs and matrices.

Theorem 6.182. *A irreducible $\iff$ $G(A)$ strongly connected.*

If A is pattern-wise symmetric, i.e., has the same zero/nonzero patterns above and below the main diagonal, then $G(A)$ is an undirected graph and the condition then becomes that $G(A)$ be connected. This provides a matrix criterion for a graph/digraph to be connected/strongly connected.

Theorem 6.183. *Let $A := A(G)$ be the adjacency matrix of a graph $G = (V, X)$ of order p. The graph (resp. digraph) G is connected (resp. strongly connected) if and only if*

$$I + A + A^2 + \cdots + A^{p-1} = C$$

has no zero entries.

The Perron-Frobenius theorem $A = [a_{ij}] \in \mathcal{M}_n(\mathbb{R})$ **nonnegative** if $a_{ij} \geq 0$ $\forall i, j = 1, \ldots, n$; $\mathbf{v} \in \mathbb{R}^n$ nonnegative if $v_i \geq 0$ $\forall i = 1, \ldots, n$. **Spectral radius** of A

$$\rho(A) = \max_{\lambda \in \mathsf{Sp}(A)} \{|\lambda|\}$$

$\mathsf{Sp}(A)$ the **spectrum** of A

Theorem 6.184 (PF – Nonnegative case)**.** *$0 \leq A \in \mathcal{M}_n(\mathbb{R})$. Then $\exists \mathbf{v} \geq \mathbf{0}$ s.t.*

$$A\mathbf{v} = \rho(A)\mathbf{v}$$

Theorem 6.185 (PF – Irreducible case)**.** *Let $0 \leq A \in \mathcal{M}_n(\mathbb{R})$ irreducible. Then $\exists \mathbf{v} > \mathbf{0}$ s.t.*

$$A\mathbf{v} = \rho(A)\mathbf{v}$$

$\rho(A) > 0$ and with algebraic multiplicity 1. No nonnegative eigenvector is associated to any other eigenvalue of A

Primitive matrices

Definition 6.186. $0 \leq A \in \mathcal{M}_n(\mathbb{R})$ ***primitive*** *(with* ***primitivity index*** $k \in \mathbb{N}_+^*$*) if* $\exists k \in \mathbb{N}_+^*$ *s.t.*

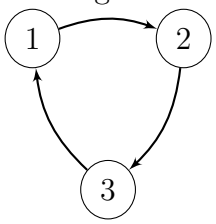
$$A^k > 0,$$

with k the smallest integer for which this is true. A ***imprimitive*** *if it is not primitive*

A primitive $\implies$ A irreducible; converse false

Theorem 6.187. *$A \in \mathcal{M}_n(\mathbb{R})$ irreducible and $\exists i = 1, \ldots, n$ s.t. $a_{ii} > 0 \implies A$ primitive*

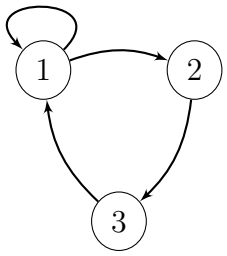
Here d is the index of imprimitivity (i.e., the number of eigenvalues that have the same modulus as $\lambda_p = \rho(A)$). If $d = 1$, then A is primitive. We have that $d = \mathsf{gcd}$ of all the lengths of closed walks in $G(A)$



Adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Closed walks in $G(A)$ (lengths): $1 \to 1$ (3), $2 \to 2$ (3), $2 \to 2$ (3) $\implies$ gcd $= 3$ $\implies$ $d = 3$ (here, all eigenvalues have modulus 1)



$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Closed walk $1 \to 1$ has length 1 $\implies$ gcd of lengths of closed walks is 1 $\implies$ A primitive

Theorem 6.188. *$0 \leq A \in \mathcal{M}_n$. A primitive $\implies$ $A^k > 0$ for some $0 < k \leq (n-1)n^n$*

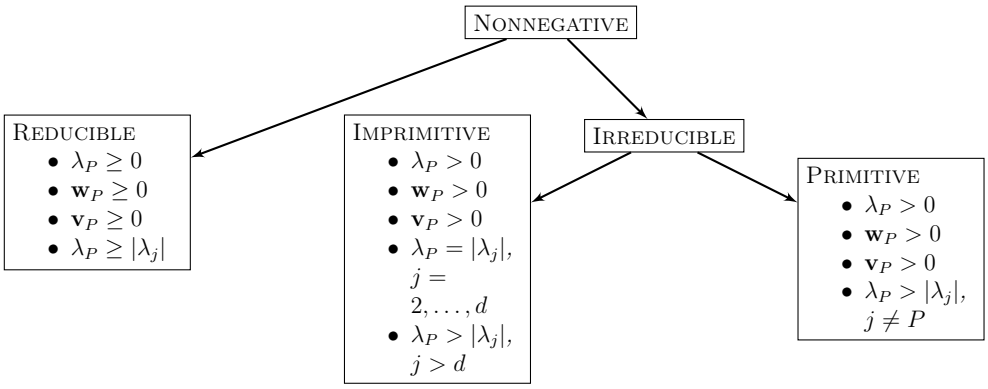
Theorem 6.189. *$A \geq 0$ primtive. Suppose the shortest simple directed cycle in $G(A)$ has length s, then primitivity index is $\leq n + s(n-1)$*

Theorem 6.190. *$0 \leq A \in \mathcal{M}_n$ primitive $\iff$ $A^{n^2-2n+2} > 0$*

Theorem 6.191. *$0 \leq A \in \mathcal{M}_n$ irreducible. A has d positive entries on the diagonal $\implies$ primitivity index $\leq 2n - d - 1$*

Theorem 6.192. *$0 \leq A \in \mathcal{M}_n$, $\lambda_P = \rho(A)$ the Perron root of A, $\mathbf{v}_P$ and $\mathbf{w}_P$ the corresponding right and left Perron vectors of A, respectively, d the index of imprimitivity of*

A (with $d = 1$ when A is primitive) and $\lambda_j \in \sigma(A)$ the spectrum of A, with $j = 2, \ldots, n$ unless otherwise specified (assuming $\lambda_1 = \lambda_P$)



Chapter 7

Quantifying graphs

Why and how to characterise a graph Graphs are everywhere! To compare graphs, understand their properties, we need ways to describe their shape and characteristics.

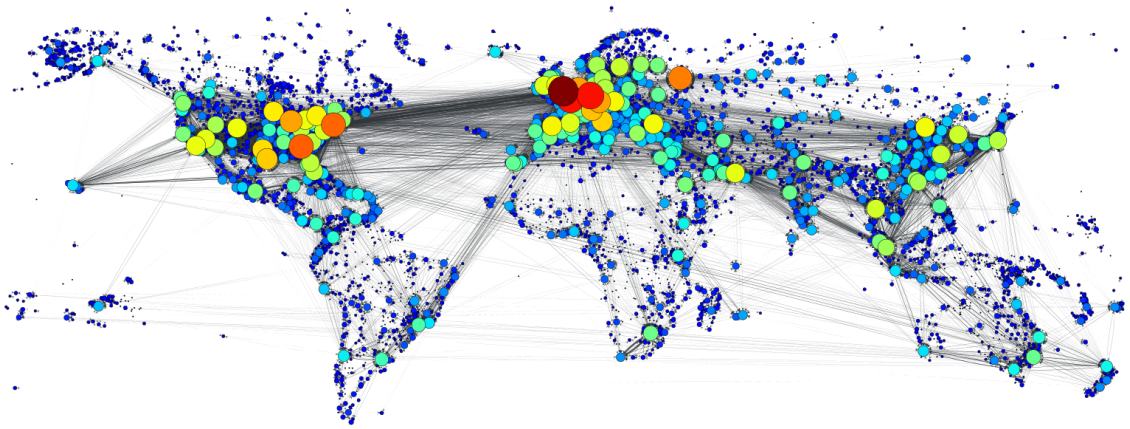


Figure 7.1: The global air transportation network. The size of the vertices indicate the degree of an airport.

Some "measures" concern the vertices, others the graph as a whole. In all that follows, unless otherwise indicated, $G = (X, A)$ is a digraph. If undirected, we write $G = (X, E)$.

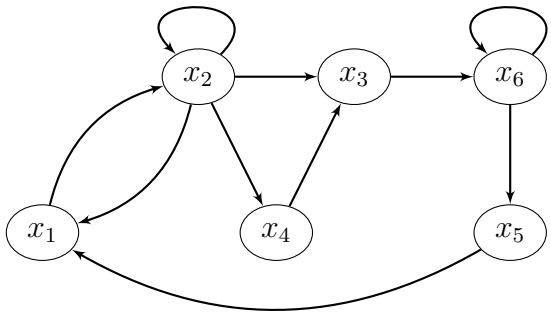
7.1 Measures specific to vertices

7.1.1 Centre of a graph

Definition 7.1 (Geodesic distance)**.** *For $x, y \in X$, the **geodesic distance** $d(x, y)$ is the length of the shortest path from x to y, with $d(x, y) = \infty$ if no such path exists*

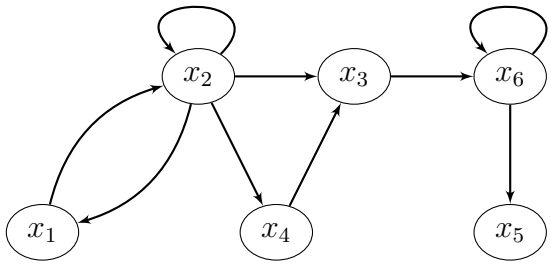
- $d(x_1, x_2) = 1$
- $d(x_1, x_3) = 2$
- $\cdots$

$$\begin{pmatrix} 0 & 1 & 2 & 2 & 4 & 3 \\ 1 & 0 & 1 & 1 & 3 & 2 \\ 3 & 4 & 0 & 5 & 2 & 1 \\ 4 & 5 & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & 4 \\ 2 & 3 & 4 & 4 & 1 & 0 \end{pmatrix}$$



- $d(x_5, x_1) = \infty$
- $d(x_3, x_1) = \infty$
- $\cdots$

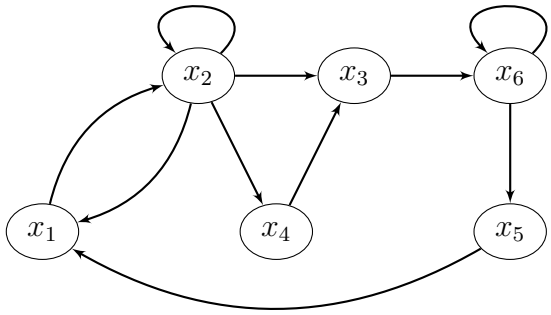
$$\begin{pmatrix} 0 & 1 & 2 & 2 & 4 & 3 \\ 1 & 0 & 1 & 1 & 3 & 2 \\ \infty & \infty & 0 & \infty & 2 & 1 \\ \infty & \infty & 1 & 0 & 3 & 2 \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$



Definition 7.2 (Vertex eccentricity)**.** *The **eccentricity** $e(x)$ of vertex $x \in X$ is*

$$e(x) = \max_{\substack{y \in X \\ y \neq x}} d(x, y)$$

$$\begin{pmatrix} 0 & 1 & 2 & 2 & \mathbf{4} & 3 \\ 1 & 0 & 1 & 1 & \mathbf{3} & 2 \\ 3 & 4 & 0 & \mathbf{5} & 2 & 1 \\ 4 & \mathbf{5} & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & \mathbf{4} \\ 2 & 3 & \mathbf{4} & \mathbf{4} & 1 & 0 \end{pmatrix}$$



Definition 7.3 (Central point)**.** *A **central point** of G is a vertex x_0 with smallest eccentricity*

Definition 7.4 (Radius). *The **radius** of G is $\rho(G) = e(x_0)$, where x_0 is a centre of G In other words,*

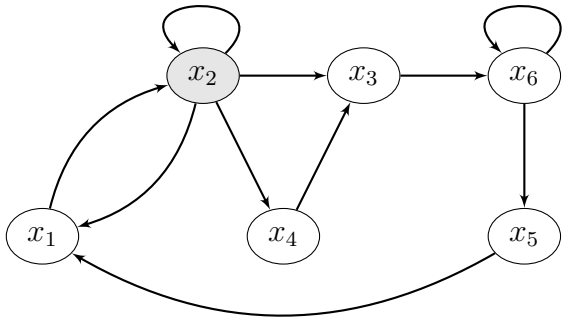
$$\rho(G) = \min_{x \in X} e(x)$$

Definition 7.5 (Centre). *The **centre** of G is the set of vertices that are central points of G, i.e.,*

$$\{x \in X : e(x) = \rho(G)\}$$

$$\begin{pmatrix} 0 & 1 & 2 & 2 & \mathbf{4} & 3 \\ 1 & 0 & 1 & 1 & \mathbf{3} & 2 \\ 3 & 4 & 0 & \mathbf{5} & 2 & 1 \\ 4 & \mathbf{5} & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & \mathbf{4} \\ 2 & 3 & \mathbf{4} & \mathbf{4} & 1 & 0 \end{pmatrix}$$

Radius is 3, x_2 is a central point (the only one) and the centre is $\{x_2\}$



7.1.2 Centrality – Betweenness and closeness

We have seen that some vertices could be labelled as being in the *centre* of a graph, others as being on the *periphery* of it. However, more generally, one can wonder about the influence of vertices. This falls under the general term of *centrality*. We have seen central vertices and vertices on the periphery, let us consider two other measures of centrality

- Betweenness centrality
- Closeness centrality

Many other forms (we will come back to this, e.g., degree centrality)

Betweenness

Definition 7.6 (Betweenness). *$G = (X, U)$ a graph, $x \in X$. The **betweenness** of v is*

$$b_{\mathcal{D}}(v) = \sum_{s \neq t \neq v \in X} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where

- *σ_{st} number of shortest geodesic paths from s to t*
- *$\sigma_{st}(v)$ number of shortest geodesic paths from s to t through v*

In other words

- For each pair of vertices (s, t), compute the shortest paths between them
- For each pair of vertices (s, t), determine the fraction of shortest paths that pass through vertex v
- Sum this fraction over all pairs of vertices (s, t)

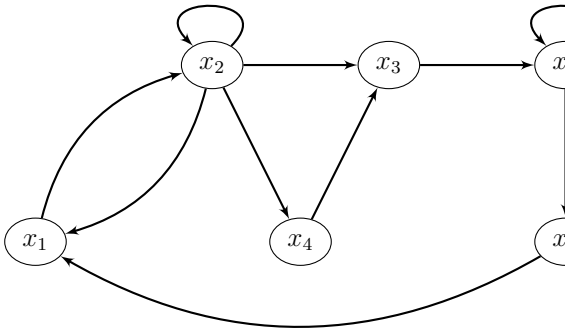
Normalising betweenness Betweenness may be normalized by dividing through the number of pairs of vertices not including v:

- for directed graphs, $(n-1)(n-2)$
- for undirected graphs, $(n-1)(n-2)/2$

Example of betweenness

`distances(G, mode="out")`

$$\begin{pmatrix} 0 & 1 & 2 & 2 & 4 & 3 \\ 1 & 0 & 1 & 1 & 3 & 2 \\ 3 & 4 & 0 & 5 & 2 & 1 \\ 4 & 5 & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & 4 \\ 2 & 3 & 4 & 4 & 1 & 0 \end{pmatrix}$$



Number of shortest paths Recall we found `distances(G, mode="out")`

$$\mathcal{D} = \begin{pmatrix} 0 & 1 & 2 & 2 & 4 & 3 \\ 1 & 0 & 1 & 1 & 3 & 2 \\ 3 & 4 & 0 & 5 & 2 & 1 \\ 4 & 5 & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & 4 \\ 2 & 3 & 4 & 4 & 1 & 0 \end{pmatrix}$$

To find the number of shortest paths between pairs of vertices, we can use powers of the adjacency matrix

Write $\mathcal{D} = [d_{ij}]$, for a given (i,j) $(i \neq j)$, if $d_{ij} = k$, then pick the (i,j) in A^k

We find

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Recall that betweenness of v is

$$b_{\mathcal{D}}(v) = \sum_{s \neq t \neq v \in X} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} (# shortest paths from s to t) is found in the matrix above

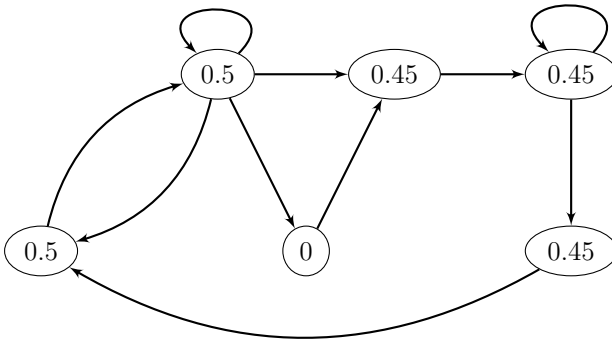
What about $\sigma_{st}(v)$, # of those shortest paths that go through v?

We can use `all_shortest_paths(G, from = s, to = t, mode = "out")`

Example of betweenness

```
betweenness(G, directed =
FALSE, normalized = TRUE)
```

Values shown in the vertices.



Closeness

Definition 7.7. *$G = (X, U)$, $v \in X$. The **closeness** of v is*

$$c_{\mathcal{D}}(v) = \frac{1}{n-1} \sum_{t \in X \setminus \{v\}} d_{\mathcal{D}}(v, t)$$

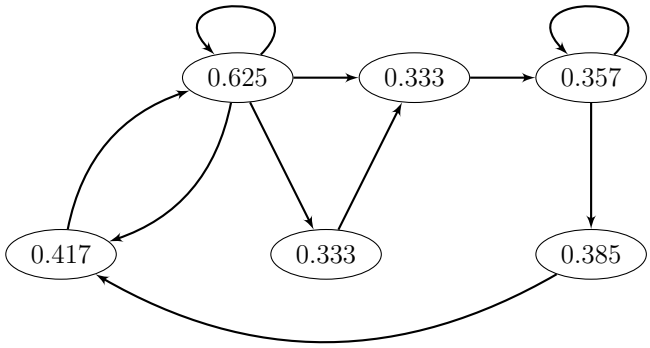
i.e., mean geodesic distance between a vertex v and all other vertices it has access to

Another definition is

$$c_{\mathcal{D}}(v) = \frac{1}{\sum_{t \in X \setminus \{v\}} d_{\mathcal{D}}(v, t)}$$

Example of (out) closeness

```
closeness(G, normalized = TRUE, mode="out")
```



7.1.3 Periphery of a graph

Diametre and periphery of a graph

Definition 7.8 (Diametre of a graph)**.** *The **diametre** of G is*

$$\delta(G) = \max_{\substack{x,y \in X \\ x \neq y}} d(x, y)$$

or, in other words,

$$\delta(G) = \max_{x \in X} e(x)$$

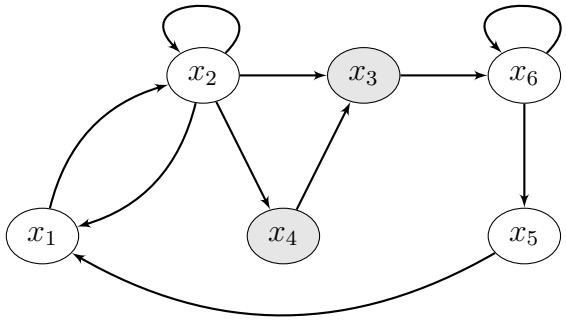
$\delta(G) < \infty \iff G$ strongly connected

Definition 7.9 (Periphery)**.** *The* ***periphery*** *of a graph is the set of vertices whose eccentricity achieves the diametre, i.e.,*

$$\{x \in X : e(x) = \delta(G)\}$$

$$\begin{pmatrix} 0 & 1 & 2 & 2 & \mathbf{4} & 3 \\ 1 & 0 & 1 & 1 & \mathbf{3} & 2 \\ 3 & 4 & 0 & \mathbf{5} & 2 & 1 \\ 4 & \mathbf{5} & 1 & 0 & 3 & 2 \\ 1 & 2 & 3 & 3 & 0 & \mathbf{4} \\ 2 & 3 & \mathbf{4} & \mathbf{4} & 1 & 0 \end{pmatrix}$$

Diametre is 5 and the periphery is $\{x_3, x_4\}$



Definition 7.10 (Antipodal vertices)**.** *A pair of vertices x and y are* ***antipodal*** *if they satisfy $d(x, y) = \delta(G)$.*

7.1.4 Degree distribution

Degree distribution

Definition 7.11 (Arc incident to a vertex)**.** *If a vertex x is the initial endpoint of an arc u, which is not a loop, the arc u is* ***incident out of vertex*** *x*

The number of arcs incident out of x plus the number of loops attached to x is denoted $d_G^+(x)$ and is the ***outer demi-degree*** *of x*

An arc ***incident into vertex*** *x and the* ***inner demi-degree*** *$d_G^-(x)$ are defined similarly*

Definition 7.12 (Degree)**.** *The* ***degree*** *of vertex x is the number of arcs with x as an endpoint, each loop being counted twice. The degree of x is denoted $d_G(x) = d_G^+(x) + d_G^-(x)$*

If each vertex has the same degree, the graph is ***regular***

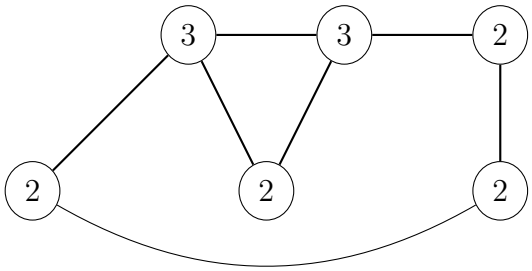
Definition 7.13 (Isolated vertex)**.** *A vertex of degree 0 is* ***isolated****.*

Definition 7.14 (Average degree of G)**.** $d(G) = \frac{1}{|V|} \sum_{v \in V} deg_G(v)$.

Definition 7.15 (Minimum degree of G)**.** $\delta(G) = \min\{deg_G(v) | v \in V\}$.

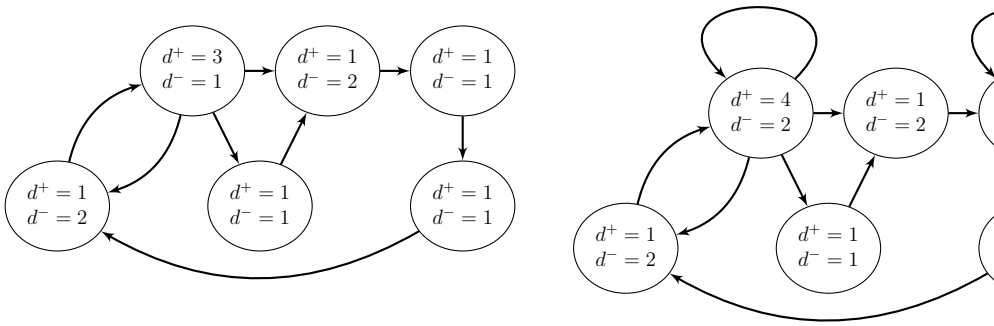
Definition 7.16 (Maximum degree of G)**.** $\Delta(G) = \max\{deg_G(v) | v \in V\}$.

Degrees in an undirected graph



Here, vertices are labelled using the degree

Degrees in a directed graph



What to consider about degrees? Degrees are often considered as a measure of popularity

Often write $k(i)$ (or k_i) for "degree of vertex i", $k^-(i)$ and $k^+(i)$ for in- and out-degree

- Minimum and maximum degree
- Minimum and maximum in/out-degree. E.g., if you consider the global air transportation network and the in/out-degree of airports, in-degree is a measure of a location's "popularity" as a travel destination
- Range of degrees in a graph: are there large discrepancies in connectivity between vertices in the graph?
- Average degree (often denoted $\langle k \rangle$ because of physicists)
- Average in/out-degree
- Variance of the degrees or in/out-degrees
- Average (nearest) neighbour degree, to encode for *preferential attachment* (one prefers to hang out with popular people)

$$k_i^{nn} = \frac{1}{k(i)} \sum_{j \in \mathcal{N}(i)} k(j)$$

or, in terms of the adjacency matrix $A = [a_{ij}]$,

$$k_i^{nn} = \frac{1}{k(i)} \sum_j a_{ij} k(j)$$

- *Excess degree*: take nearest neighbour degree but do not consider the edge/arc followed to get to the neighbour

- Degree, nearest neighbour and excess degree distributions

Degrees in `igraph`

- `degree` gives the degrees of the vertices
- `degree_distribution` gives numeric vector of the same length as the maximum degree plus one. The first element is the relative frequency zero degree vertices, the second vertices with degree one, etc.
- `knn` calculate the average nearest neighbor degree of the given vertices and the same quantity in the function of vertex degree
- `strength` sums up the edge weights of the adjacent edges for each vertex

Degree from adjacency matrix Suppose adjacency matrix take the form $A = [a_{ij}]$ with $a_{ij} = 1$ if there is an arc from the vertex indexed i to the vertex indexed j and 0 otherwise. (Could be the other way round, using A^T, just make sure)

Let $\mathbf{e} = (1, \ldots, 1)^T$ be the vector of all ones
$A\mathbf{e} = (d_G^+(1), \ldots, d_G^+(1))^T$ (out-degree)
$\mathbf{e}^T A = (d_G^-(1), \ldots, d_G^-(1))$ (in-degree)

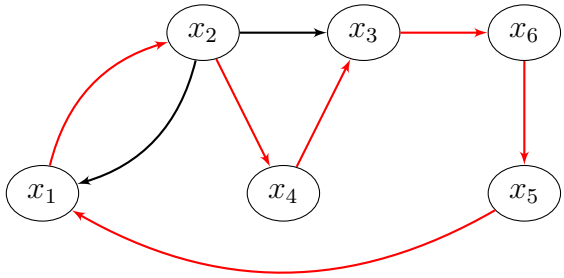
7.2 Measures at the graph level

7.2.1 Circumference & Girth

Circumference

Definition 7.17 (Circumference)**.** *In an undirected (resp. directed) graph, the total number of edges (resp. arcs) in the longest cycle of graph G is the* ***circumference*** *of G*

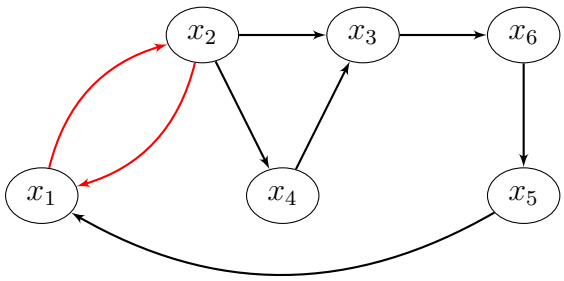
Circumference is 6.



Girth

Definition 7.18 (Girth)**.** *The total number of edges in the shortest cycle of graph G is the* ***girth*** *$g(G)$*

Girth is 2.



7.2.2 Graph density

Completeness

Definition 7.19 (Complete undirected graph)**.** *An undirected graph is complete if every two of its vertices are adjacent.*

Definition 7.20 (Complete digraph)**.** *A digraph $D(V, A)$ is complete if $\forall u, v \in V$, $uv \in A$.*

In case of simple graphs, completeness effectively means that "information" can be transmitted from every vertex to every other vertex quickly (1 step)

It can be useful to know how far away we are from being complete

Number of edges/arcs in a complete graph $G = (X, E)$ undirected and simple of order n has at most

$$\frac{n(n-1)}{2}$$

edges, while $G = (X, A)$ directed and simple of order n has at most

$$n(n-1)$$

arcs

Density of a graph

Definition 7.21 (Density)**.** *The fraction of maximum number of edges or arcs present in the graph is the* ***density*** *of the graph.*

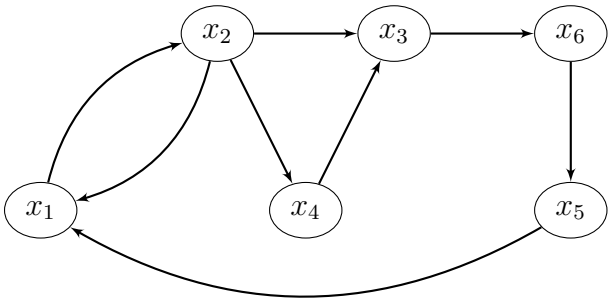
If the graph has p edges or arcs, then its density is, respectively,

$$\frac{2p}{n(n-1)}$$

or

$$\frac{p}{n(n-1)}$$

Example of density



Graph has order 6
thus a max of 30
Here, 8 arcs $\implies$ d
0.267
(26.7% of arcs are pr

7.2.3 Graph connectivity

Connectedness We have already seen connectedness (quasi- or strong in the oriented case)

Connectedness is important in terms of characteristing graph properties, as it shows the capacity of the graph to convey information to all the members of the graph (the vertices)

Definition 7.22 (Connected graph)**.** *A **connected graph** is a graph that contains a chain $\mu[x, y]$ for each pair x, y of distinct vertices*

Denote $x \equiv y$ the relation "$x = y$, or $x \neq y$ and there exists a chain in G connecting x and y". $\equiv$ is an equivalence relation since

1. $x \equiv y$ [reflexivity]
2. $x \equiv y \implies y \equiv x$ [symmetry]
3. $x \equiv y, y \equiv z \implies x \equiv z$ [transitivity]

Definition 7.23 (Connected component of a graph)**.** *The classes of the equivalence relation $\equiv$ partition X into connected sub-graphs of G called **connected components***

Articulation set

Definition 7.24 (Articulation set)**.** *For a connected graph, a set A of vertices is called an **articulation set** (or a **cutset**) if the subgraph of G generated by $X - A$ is not connected*

`articulation_points(G)` in `igraph` (assumes the graph is undirected, makes it so if not)

Strongly connected graphs $G = (X, U)$ connected. A **path of length 0** is any sequence $\{x\}$ consisting of a single vertex $x \in X$

For $x, y \in X$, let $x \equiv y$ be the relation "there is a path $\mu_1[x, y]$ from x to y as well as a path $\mu_2[y, x]$ from y to x". This is an equivalence relation (it is reflexive, symmetric and transitive)

Definition 7.25 (Strong components)**.** *Sets of the form*

$$A(x_0) = \{x : x \in X, x \equiv x_0\}$$

*are equivalence classes; they partition X and are the **strongly connected components** of G*

Definition 7.26 (Strongly connected graph)**.** *G **strongly connected** if it has a single strong component*

Definition 7.27 (Minimally connected graph)**.** *G is **minimally connected** if it is strongly connected and removal of any arc destroys strong-connectedness*

Definition 7.28 (Contraction)**.** *$G = (X, U)$. The **contraction** of the set $A \subset X$ of vertices consists in replacing A by a single vertex a and replacing each arc into (resp. out of) A by an arc with same index into (resp. out of) a*

Quasi-strong connectedness

Definition 7.29 (Quasi-strong connectedness)**.** *G **quasi-strongly connected** if $\forall x, y \in X$, exists $z \in X$ (denoted $z(x, y)$ to emphasize dependence on x, y) from which there is a path to x and a path to y*

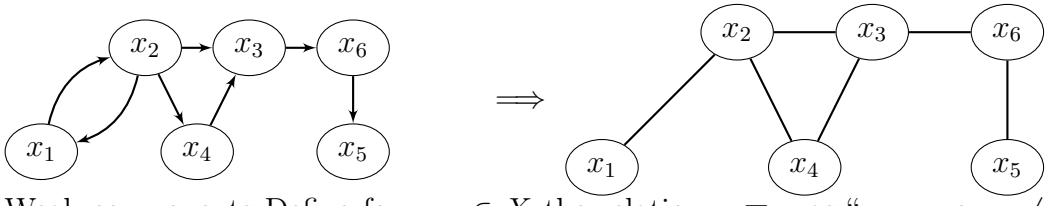
Strongly connected $\implies$ quasi-strongly connected (take $z(x, y) = x$); converse not true

Quasi-strongly connected $\implies$ connected

Lemma 7.30. *$G = (X, U)$ has a root $\iff$ G quasi-strongly connected*

Weak-connectedness

Definition 7.31 (Weakly connected graph)**.** *$G = (X, U)$ **weakly connected** if $G = (X, E)$ connected, where E is obtained from U by ignoring the direction of arcs*



Weak components Define for $x, y \in X$ the relation $x \equiv y$ as "$x = y$ or $x \neq y$ and there is a chain in G connecting x and y" [like for components in an undirected graph, except the graph is directed here]

This defines an equivalence relation

Definition 7.32 (Weak components)**.** *Sets of the form*

$$A(x_0) = \{x : x \in X, x \equiv x_0\}$$

*are equivalence classes partitioning X into the **weakly connected components** of G*

$G = (X, U)$ is weakly connected if there is a single weak component

Components in `igraph`

- `is_connected` decides whether the graph is weakly or strongly connected
- `components` finds the maximal (weakly or strongly) connected components of a graph
- `count_components` does almost the same as components but returns only the number of clusters found instead of returning the actual clusters
- `component_distribution` creates a histogram for the maximal connected component sizes
- `decompose` creates a separate graph for each component of a graph
- `subcomponent` finds all vertices reachable from a given vertex, or the opposite: all vertices from which a given vertex is reachable via a directed path

7.2.4 Cliques

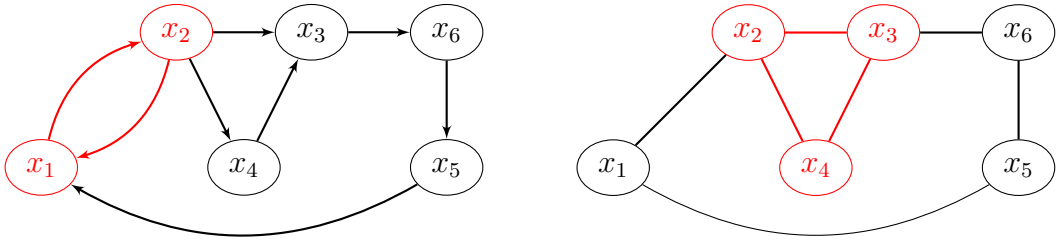
Cliques

Definition 7.33 (Clique in undirected graphs)**.** *$G = (X, E)$ a simple undirected graph. A **clique** is a subgraph G' of G such that all vertices in G' are adjacent*

Definition 7.34 (n-clique)**.** *A simple, complete graph on n vertices is called an n-**clique** and is often denoted K_n*

Definition 7.35 (Clique in directed graphs)**.** *$G = (X, U)$ a simple directed graph. A **clique** is a subgraph G' of G such that all vertices in G' are mutually adjacent*

Definition 7.36 (Maximal clique)**.** *A **maximal clique** is a clique that cannot be extended by adding another adjacent vertex*



Cliques in `igraph`

- `cliques` find all complete subgraphs in the input graph, obeying the size limitations given in the min and max arguments
- `largest_cliques` finds all largest cliques in the input graph
- `max_cliques` finds all maximal cliques in the input graph (The largest cliques are always maximal, but a maximal clique is not necessarily the largest)
- `count_max_cliques` counts the maximal cliques
- `clique_num` calculates the size of the largest clique(s)

7.2.5 k-cores

k-core

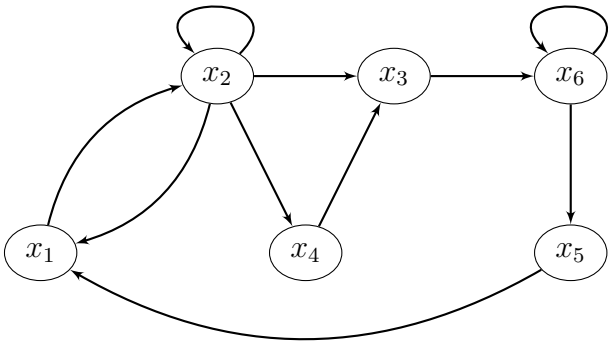
Definition 7.37 (k-core of a graph)**.** *$G = (X, U)$ a graph. The **k-core** of G is a maximal subgraph in which each vertex has degree at least k*

Definition 7.38 (Coreness of a vertex)**.** *$G = (X, U)$ a graph, $x \in X$. The **coreness** of x is k if x belongs to the k-core of G but not to the $k + 1$ core of G*

For directed graphs, in-cores or out-cores depending on whether in-degree or out-degree is used

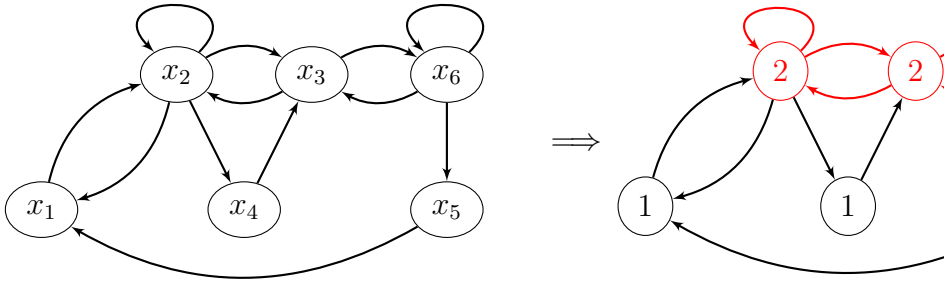
In `igraph`: `coreness`

Coreness in the directed case

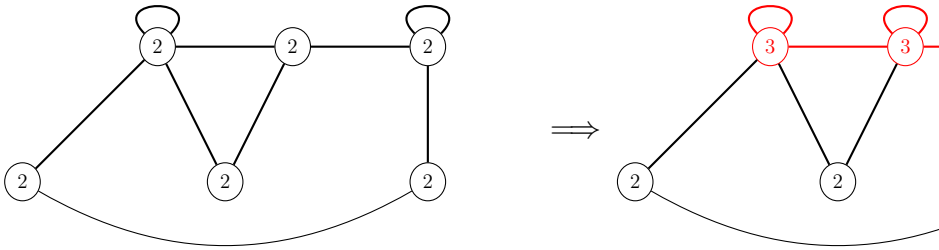


G has only a 1-in-core and 1-out-core: there is no (maximal) subgraph in which the in- or out-degree is larger than 1

In-coreness in the directed case



Coreness in the undirected case



Chapter 8

The PageRank algorithm

What makes an important webpage? In days of yore, the web was a small thing

Alta Vista was the search engine of choice

Google started in 1998, based on an algorithm (PageRank) described in a paper of Page, Brin, Motwani and Winograd

Overview Give each page a rating (of its importance), a recursively defined measure whereby a page becomes important if important pages link to it

Recursive definition: the importance of a page refers back to the importance of other pages that link to it

Random surfer model: a random surfer on the web follows links from page to page. Page rank $\simeq \mathbb{P}$ random surfer lands on a particular page. Popular page $\implies$ higher probability to go there

Example of a Markov chain

8.1 Markov chains

Markov chain A Markov chain is a *stochastic process* in which the evolution through time depends only on the current state of the system (we say the process is *memoryless*)

Markov chains are an interesting combination of matrix theory and graph theory

They form the theoretical foundation for Hidden Markov processes or Markov Chain Monte Carlo (MCMC) methods, are used in ML

Conduct an experiment with a set of r possible outcomes

$$S = \{S_1, \ldots, S_r\}$$

Experiment repeated n times (with n large, potentially infinite)

System has *no memory*: the next state depends only on the present state

Probability of S_j occurring on the next step, given that S_i occurred on the last step, is

$$p_{ij} = p(S_j|S_i)$$

Suppose that S_i is the current state, then one of $S_1, \ldots, S_r$ must be the next state; so

$$p_{i1} + p_{i2} + \cdots + p_{ir} = 1, \quad 1 \le i \le r$$

(Some of the p_{ij} can be zero, all that is needed is that $\sum_{j=1}^{r} p_{ij} = 1$ for all i)

Definition 8.1. *An experiment with finite number of possible outcomes $S_1, \ldots, S_r$ is repeated. The sequence of outcomes is a* ***Markov chain*** *if there is a set of r^2 numbers $\{p_{ij}\}$ such that the conditional probability of outcome S_j on any experiment given outcome S_i on the previous experiment is p_{ij}, i.e., for $1 \le i, j \le r$, $n = 1, \ldots,$*

$$p_{ij} = \mathsf{Pr}(S_j \text{ on experiment } n+1 \mid S_i \text{ on experiment } n)$$

Outcomes $S_1, \ldots, S_r$ are ***states*** *and p_{ij} are* ***transition probabilities****. $P = [p_{ij}]$ the* ***transition matrix***

The matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ & & & \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix}$$

has

- nonnegative entries, $p_{ij} \ge 0$
- entries less than 1, $p_{ij} \le 1$
- row sum 1, which we write

$$\sum_{j=1}^{r} p_{ij} = 1, \quad i = 1, \ldots, r$$

or, using the notation $\mathbb{1}^T = (1, \ldots, 1)$,

$$P\mathbb{1} = \mathbb{1}$$

8.2 Running example – Mendelian inheritance

(super simple) Mendelian genetics A *phenotypic trait* (eye colour, hair colour, etc.) is determined by a specific pair of alleles, each of which may be two types, say G and g

Each individual can have

- GG combination (*dominant*)
- Gg or gG, considered equivalent genetically (*hybrid*)
- gg combination (*recessive*)

Individuals bearing GG or gg alleles are *homozygotes*, hybrids with Gg alleles are called *heterozygotes*

GG and gg combinations lead to different phenotypes, Gg combination leads to expressing the same phenotype as individuals bearing a GG combination, hence the name dominant given to GG

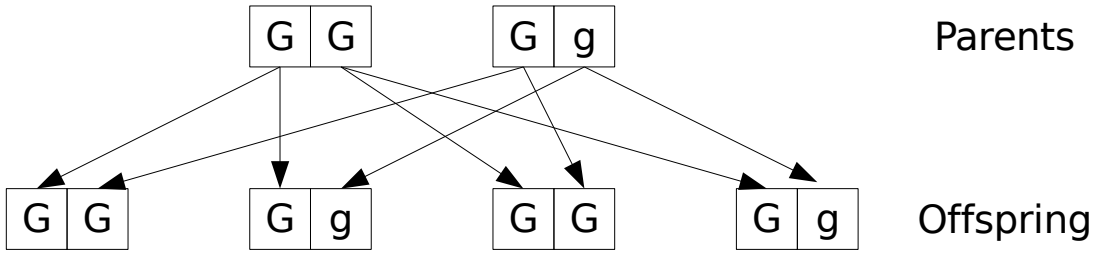
In sexual reproduction, offspring inherit one allele of the pair from each parent

Alleles inherited from each parent are selected at random, independently of each other

This determines probability of occurrence of each type of offspring. The offspring

- of two GG parents must be GG
- of two gg parents must be gg
- of one GG and one gg parent must be Gg
- other cases must be examined in more detail

GG and Gg parents Suppose one parent GG and the other Gg



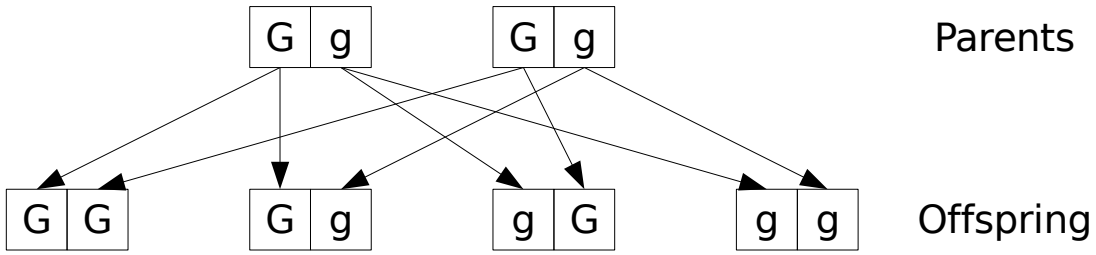
		Parent 1	
		G	G
Parent 2	G	GG	GG
	g	Gg	Gg

To determine $\mathbb{P}$ that offspring is of a certain type, count number of outcomes of each type (GG and Gg) and divide by 4

$\implies$ offspring have probability

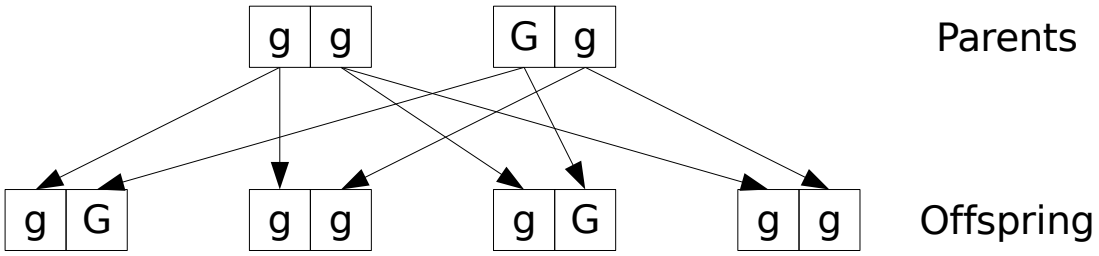
- 1/2 of being GG
- 1/2 of being Gg

Gg and Gg parents Both parents are hybrid



$\implies$ offspring have probability
- 1/4 of being GG
- 1/2 of being Gg
- 1/4 of being gg

gg and Gg parents Recessive and hybrid parents



$\implies$ offspring have probability
- 1/2 of being Gg
- 1/2 of being gg

8.3 Repetition of the process

General case $p_i(n)$: probability that state S_i occurs on the n^{th} repetition of the experiment, $1 \leq i \leq r$

Since one the states S_i must occur on the n^{th} repetition

$$p_1(n) + p_2(n) + \cdots + p_r(n) = 1$$

$p_i(n+1)$: probability that state S_i, $1 \leq i \leq r$, occurs on $(n+1)^{th}$ repetition of the experiment

r ways to be in state S_i at step $n+1$:

1. Step n is S_1. Probability of getting S_1 on n^{th} step is $p_1(n)$, and probability of having S_i after S_1 is p_{1i}. Therefore $P(S_i|S_1) = p_{1i}p_1(n)$
2. We get S_2 on step n and S_i on step $(n+1)$. Then $P(S_i|S_2) = p_{2i}p_2(n)$

..

r. Probability of occurrence of S_i at step $n+1$ if S_r at step n is $P(S_i|S_r) = p_{ri}p_r(n)$

$$\begin{aligned} \implies p_i(n+1) &= P(S_i|S_1) + \cdots + P(S_i|S_r) \\ &= p_{1i}p_1(n) + \cdots + p_{ri}p_r(n) \end{aligned}$$

Therefore,

$$\begin{aligned} p_1(n+1) &= p_{11}p_1(n) + p_{21}p_2(n) + \cdots + p_{r1}p_r(n) \\ &\vdots \\ p_r(n+1) &= p_{1r}p_1(n) + p_{2r}p_2(n) + \cdots + p_{rr}p_r(n) \end{aligned}$$

In matrix form

$$p(n+1) = p(n)P, \quad n = 1, 2, 3, \ldots$$

where $p(n) = (p_1(n), p_2(n), \ldots, p_r(n))$ is a (row) probability vector and $P = (p_{ij})$ is a $r \times r$ *transition matrix*,

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ & & & \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix}$$

So

$$(p_1(n+1), \ldots, p_r(n+1)) = (p_1(n), \ldots, p_r(n)) \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ & & & \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix}$$

Easy to check that this gives the same expression as before

Stochastic matrices

Definition 8.2 (Stochastic matrix)**.** *The nonnegative $r \times r$ matrix M is **stochastic** if $\sum_{j=1}^{r} a_{ij} = 1$ for all $i = 1, 2, \ldots, r$*

If each row sum and each column sum equals one, the matrix is **doubly stochastic**

Theorem 8.3. *Let M be a stochastic matrix. Then all eigenvalues λ of M are such that $|\lambda| \leq 1$. Furthermore, $\lambda = 1$ is an eigenvalue of M*

Long time behaviour Let $p(0)$ be the initial distribution (row) vector. Then

$$\begin{aligned} p(1) &= p(0)P \\ p(2) &= p(1)P \\ &= (p(0)P)P \\ &= p(0)P^2 \end{aligned}$$

Iterating, we get, for any n,

$$p(n) = p(0)P^n$$

Therefore,

$$\begin{aligned} \lim_{n \to +\infty} p(n) &= \lim_{n \to +\infty} p(0)P^n \\ &= p(0) \lim_{n \to +\infty} P^n \end{aligned}$$

if this limit exists

$$\lim_{n \to +\infty} p(n) = p(0) \lim_{n \to +\infty} P^n$$

Does the limit exist?

Theorem 8.4. *If M, N are nonsingular stochastic matrices, then MN is a stochastic matrix*

So the product of any number of stochastic matrices is also stochastic

Corollary 8.5. *If M is a nonsingular stochastic matrix, then for any $k \in \mathbb{N}$, M^k is a stochastic matrix*

8.4 Regular Markov chains

Definition 8.6 (Regular Markov chain)**.** *A **regular** Markov chain has P^k (entry-wise) positive for some integer $k > 0$, i.e., P^k has only positive entries*

Definition 8.7 (Primitive matrix)**.** *A nonnegative matrix M is **primitive** if, and only if, there is an integer $k > 0$ such that M^k is positive.*

Theorem 8.8. *Markov chain regular $\iff$ transition matrix P primitive*

Behaviour of a regular MC

Theorem 8.9. *If P is the transition matrix of a regular Markov chain, then*

1. *the powers P^n approach a stochastic matrix W*
2. *each row of W is the same (row) vector $w = (w_1, \ldots, w_r)$*
3. *the components of w are positive*

So if the Markov chain is regular

$$\lim_{n \to +\infty} p(n) = p(0) \lim_{n \to +\infty} P^n = p(0)W$$

Computing W If $p(n)$ converges, then $p(n+1) = p(n)P$, so w is a **fixed point** of the system. Write

$$wP = w$$

and solve for w, i.e., find w as left eigenvector corresponding to the eigenvalue 1 or as (right) eigenvector associated to eigenvalue 1 for the transpose of P

$$P^T w^T = w^T$$

w might have to be normalized (you want a probability vector). Check that the norm $\|w\|$ defined by

$$\|w\| = w_1 + \cdots + w_r$$

is equal to one. If not, use

$$\frac{w}{\|w\|}$$

8.5 Back to the genetics example

Suppose we want to understand what it means to have hybrid individuals in the population

Investigate this using a process of continued matings

- Start with an individual of known or unknown genetic character (dominant, hybrid or recessive) and mate it with a hybrid
- Assume that the mating results in at least one offspring; choose one of the offspring at random and mate it with a hybrid
- Repeat this process through a number of generations

What can we expect in terms of the genetic composition of the population after a while?
$\Longrightarrow$ consider MC with states GG, Gg and gg

3 states: $S_1 = \text{GG}$, $S_2 = \text{Gg}$ and $S_3 = \text{gg}$; we use GG, Gg and gg as well to name the states

$\nearrow$	GG	Gg	gg
GG	0.5	0.5	0
Gg	0.25	0.5	0.25
gg	0	0.5	0.5

The transition probabilities are thus

$$P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

$$P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

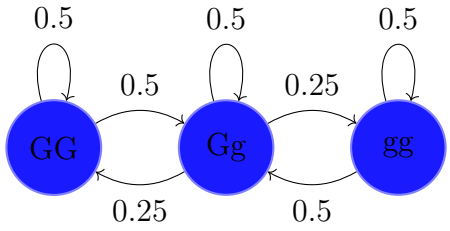
so

$$P^2 = \begin{pmatrix} \frac{3}{8} & \frac{1}{2} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{2} & \frac{3}{8} \end{pmatrix}$$

$\Longrightarrow$ P primitive $\Longrightarrow$ Markov chain regular

Theorem 8.10. *M primitive if the associated connection graph is strongly connected* and *there is at least one positive entry on the diagonal of M*

This is checked directly on the transition graph



Compute left eigenvector associated to 1

$$(w_1, w_2, w_3) \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} = (w_1, w_2, w_3)$$

$$\begin{aligned} \frac{1}{2}w_1 + \frac{1}{4}w_2 &= w_1 \\ \frac{1}{2}w_1 + \frac{1}{2}w_2 + \frac{1}{2}w_3 &= w_2 \\ \frac{1}{4}w_2 + \frac{1}{2}w_3 &= w_3 \end{aligned}$$

So $w_1 = w_2/2$, $w_3 = w_2/2$ and thus

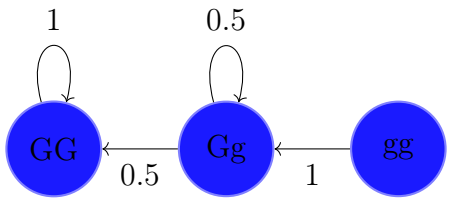
$$\frac{1}{4}w_2 + \frac{1}{2}w_2 + \frac{1}{4}w_2 = w_2,$$

that is, $w_2 = w_2$, i.e., w_2 can take any value

$$\Longrightarrow \quad w = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$$

8.6 Changing the setting of the genetic experiment

Suppose now the same type of experiment, but mate each new generation with a GG individual instead of a Gg individual



↗	GG	Gg	gg
GG	1	0	0
Gg	0.5	0.5	0
gg	0	1	0

$$P = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

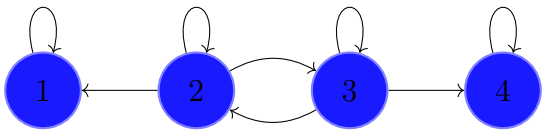
- leave gg after 1 iteration and can never return
- when we leave Gg, we can never return
- we can never leave GG when we get there

8.7 Absorbing Markov chains

Definition 8.11 (Absorbing state)**.** *A state S_i in a Markov chain is **absorbing** if whenever it occurs on the n^{th} generation of the experiment, it then occurs on every subsequent step. In other words, S_i is absorbing if $p_{ii} = 1$ and $p_{ij} = 0$ for $i \neq j$*

Definition 8.12 (Absorbing chain)**.** *A Markov chain is **absorbing** if it has at least one absorbing state, and if from every state it is possible to go to an absorbing state. In an absorbing Markov chain, a state that is not absorbing is called **transient***

Suppose we have a chain like the following



1. Does the process eventually reach an absorbing state?
2. What is the average number of steps spent in a transient state, if starting in a transient state?
3. What is the average number of steps before entering an absorbing state?
4. What is the probability of being absorbed by a given absorbing state, when there are more than one, when starting in a given transient state?

The answer to the first question ("Does the process eventually reach an absorbing state?") is given by the following result

Theorem 8.13. *In an absorbing Markov chain, the probability of reaching an absorbing state is 1*

To answer the other questions, write the transition matrix in **standard** form

For an absorbing chain with k absorbing states and $r - k$ transient states, write transition matrix as

$$P = \begin{pmatrix} \mathbb{I}_k & \mathbf{0} \\ R & Q \end{pmatrix}$$

with following meaning

	Absorbing states	Transient states
Absorbing states	$\mathbb{I}_k$	$\mathbf{0}$
Transient states	R	Q

with $\mathbb{I}_k$ the $k \times k$ identity matrix, $\mathbf{0}$ an $k \times (r-k)$ matrix of zeros, R an $(r-k) \times k$ matrix and Q an $(r-k) \times (r-k)$ matrix. The matrix $\mathbb{I}_{r-k} - Q$ is invertible. Let

- $N = (\mathbb{I}_{r-k} - Q)^{-1}$ the **fundamental matrix** of the MC
- T_i sum of the entries on row i of N
- $B = NR$

Answers to our remaining questions:

2. N_{ij} average number of times the process is in the jth transient state if it starts in the ith transient state
3. T_i average number of steps before the process enters an absorbing state if it starts in the ith transient state
4. B_{ij} probability of eventually entering the jth absorbing state if the process starts in the ith transient state

Back to the genetic example The matrix is already in standard form

$$P = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \mathbb{I}_1 & \mathbf{0} \\ R & Q \end{pmatrix}$$

with $\mathbb{I}_1 = 1$, $\mathbf{0} = (0 \;\; 0)$ and

$$R = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} \qquad Q = \begin{pmatrix} \frac{1}{2} & 0 \\ 1 & 0 \end{pmatrix}$$

We have

$$\mathbb{I}_2 - Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{2} & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{pmatrix}$$

so

$$N = (\mathbb{I}_2 - Q)^{-1} = 2 \begin{pmatrix} 1 & 0 \\ 1 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix}$$

Then

$$T = N\mathbb{1} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

and

$$B = NR = \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

2. N_{ij} average number of times the process is in the jth transient state if it starts in the ith transient state

$$N = \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix}$$

3. T_i average number of steps before the process enters an absorbing state if it starts in the ith transient state

$$T = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

4. B_{ij} probability of eventually entering the jth absorbing state if the process starts in the ith transient state

$$B\begin{pmatrix}1\\1\end{pmatrix}$$

Appendix A

Review/presentation of some required concepts

In this course, we rely on notions you acquired in first year Linear Algebra. So let us (briefly) go over material in these courses. I also add (for some of you) a few things that will be handy and establish some terminology that we use throughout the course. Some of the results in this appendix will be studied and proved in class, others you can just admit.

A.1 Sets

A.1.1 Sets and elements

Sets are some of the most elementary structures used in mathematics. They are also extremely useful in programming languages.

Definition A.1 (Set)**.** *A **set** X is a collection of **elements**.*

We write $x \in X$ or $x \notin X$ to indicate that the element x belongs to the set X or does not belong to the set X, respectively.

Definition A.2 (Subset)**.** *Let X be a set. The set S is a **subset** of X, which is denoted $S \subseteq X$, if all its elements belong to X. We say that the subset S is a **proper subset** of X and write $S \subsetneq X$, if it is a subset of X and not equal to X.*

Let A and B two sets. U the universal set ($A \subseteq U$ and $B \subseteq U$).

Definition A.3. *Union of A and B $A \cup B = \{x : x \in A \text{ or } x \in B\}$ set consisting of elements belonging to A or B*

Definition A.4 (Intersection of A and B)**.** $A \cap B = \{x : x \in A \text{ and } x \in B\}$ *set containing those elements belonging to both A and B*

Definition A.5 (Empty set)**.** *The empty set $\emptyset$ contains no element.*

Definition A.6 (Disjoints sets)**.** *If $A \cap B = \emptyset$, then A and B are disjoints sets. (A and B have no elements in common).*

Definition A.7 (Complement of A)**.** $\bar{A} = \{x : x \in U \text{ and } x \notin A\}$ *set consisting of all elements of U not belonging to A.*

Definition A.8 (Difference)**.** $A \backslash B = \{x : x \in A \text{ and } x \notin B\}$ *set containing those elements of A that do not belong to B*

Definition A.9 (Partition)**.** *Let A be a non-empty set. A partition of A is the set $\{A_1, A_2, \ldots, A_n\}$ such that*

- $\forall i \in \{1, \ldots, n\}$, $A_i \neq \emptyset$ *and* $A_i \subset A$ *(non-empty and subset of A)*
- *If* $A_i \neq A_j$ *then* $A_i \cap A_j = \emptyset$ *(every two subsets are disjoint)*
- $A_1 \cup A_2 \cup \cdots \cup A_n = A$

A.1.2 Quantifiers

A shorthand notation for "for all elements x belonging to X" is $\forall x \in X$. For example, if $X = \mathbb{R}$, the *field* of real numbers, then $\forall x \in \mathbb{R}$ means "for all real numbers x".

A shorthand notation for "there exists an element x in the set X" is $\exists x \in X$. $\forall$ and $\exists$ are **quantifiers**.

A.1.3 Intersection and union of sets

Let X and Y be two sets.

Definition A.10 (Intersection)**.** *The* ***intersection*** $X \cap Y$ *of X and Y is the set of elements that belong to X* ***and*** *to Y,*

$$X \cap Y = \{x : x \in X \ \textbf{\textit{and}} \ x \in Y\}.$$

Definition A.11 (Union)**.** *The* ***union*** $X \cup Y$ *of X and Y is the set of elements that belong to X* ***or*** *to Y,*

$$X \cup Y = \{x : x \in X \ \textbf{\textit{or}} \ x \in Y\}.$$

In mathematics, or=and/or in common parlance and I tend to get angry when the latter formulation is used. If you want to specify that elements should belong to one of the sets but not to both, the notion of **exclusive or** (xor) also exists, which is defined as

$$(X \cup Y) \setminus (X \cap Y).$$

A.2 Just enough logic to get by

A **proposition** is an assertion (or statement) whose truth value (true or false) can be asserted. For example, a theorem is a proposition that has been shown to be true. "The sky is blue" is also a proposition. Let A be a proposition. We generally write

$$A$$

to mean that A is true, and

$$\textbf{not } A$$

to mean that A is false. **not** A is the **negation** of A (or **not** A is the negative of A).

Let A, B be propositions. Then

- $A \Rightarrow B$ (read A implies B) means that whenever A is true, then so is B.
- $A \Leftrightarrow B$, also denoted A if and only if B (A iff B for short), means that $A \Rightarrow B$ **and** $B \Rightarrow A$
  We also say that A and B are **equivalent**.

Let A and B be propositions. Then

$$(A \Rightarrow B) \Leftrightarrow (\textbf{not } B \Rightarrow \textbf{not } A).$$

Necessary or sufficient conditions Suppose we want to establish whether a given statement P is true, depending on the truth value of a statement H. Then we say that

- H is a **necessary condition** if $P \Rightarrow H$
  (It is necessary that H be true for P to be true; so whenever P is true, so is H)
- H is a **sufficient condition** if $H \Rightarrow P$
  (It suffices for H to be true for P to also be true)
- H is a **necessary and sufficient condition** if $H \Leftrightarrow P$, i.e., H and P are equivalent

Playing with quantifiers For the quantifiers $\forall$ (for all) and $\exists$ (there exists),

$$\exists \text{ is the contraposite of } \forall$$

Therefore, for example, the contraposite of

$$\forall x \in X, \exists y \in Y$$

is

$$\exists x \in X, \forall y \in Y$$

A.3 Vectors and vector spaces

A.3.1 Vectors

A **vector** $\mathbf{v}$ is an ordered n-tuple of real or complex numbers

Denote $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ (real or complex numbers). For $v_1, \ldots, v_n \in \mathbb{F}$,

$$\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}^n$$

is a vector. $v_1, \ldots, v_n$ are the **components** of $\mathbf{v}$

If unambiguous, we write v. Otherwise, $\mathbf{v}$ or $\vec{v}$

A.3.2 Vector space

Definition A.12 (Vector space)**.** *A **vector space** over $\mathbb{F}$ is a set V together with two binary operations, **vector addition**, denoted $+$, and **scalar multiplication**, that satisfy the relations:*

1. $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$, $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
2. $\forall \mathbf{v}, \mathbf{w} \in V$, $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$
3. $\exists \mathbf{0} \in V$, *the zero vector, such that* $\mathbf{v} + \mathbf{0} = \mathbf{v}$ *for all* $\mathbf{v} \in V$
4. $\forall \mathbf{v} \in V$, *there exists an element* $\mathbf{w} \in V$, *the additive inverse of* $\mathbf{v}$, *such that* $\mathbf{v} + \mathbf{w} = \mathbf{0}$
5. $\forall \alpha \in \mathbb{R}$ *and* $\forall \mathbf{v}, \mathbf{w} \in V$, $\alpha(\mathbf{v} + \mathbf{w}) = \alpha\mathbf{v} + \alpha\mathbf{w}$
6. $\forall \alpha, \beta \in \mathbb{R}$ *and* $\forall \mathbf{v} \in V$, $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$
7. $\forall \alpha, \beta \in \mathbb{R}$ *and* $\forall \mathbf{v} \in V$, $\alpha(\beta\mathbf{v}) = (\alpha\beta)\mathbf{v}$
8. $\forall \mathbf{v} \in V$, $1\mathbf{v} = \mathbf{v}$

A.3.3 Norms

Definition A.13 (Norm)**.** *Let V be a vector space over $\mathbb{F}$, and $\mathbf{v} \in V$ be a vector. The **norm** of $\mathbf{v}$, denoted $\|\mathbf{v}\|$, is a function from V to $\mathbb{R}_+$ that has the following properties:*

1. *For all $\mathbf{v} \in V$, $\|\mathbf{v}\| \geq 0$ with $\|\mathbf{v}\| = 0$ iff $\mathbf{v} = \mathbf{0}$.*
2. *For all $\alpha \in \mathbb{F}$ and all $\mathbf{v} \in V$, $\|\alpha\mathbf{v}\| = |\alpha| \, \|\mathbf{v}\|$.*
3. *For all $\mathbf{u}, \mathbf{v} \in V$, $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$.*

Let V be a vector space (for example, $\mathbb{R}^2$ or $\mathbb{R}^3$)

The **zero element** (or **zero vector**) is the vector $\mathbf{0} = (0, \ldots, 0)$

The **additive inverse** of $\mathbf{v} = (v_1, \ldots, v_n)$ is $-\mathbf{v} = (-v_1, \ldots, -v_n)$

For $\mathbf{v} = (v_1, \ldots, v_n) \in V$, the length (or Euclidean norm) of $\mathbf{v}$ is the **scalar**

$$\|\mathbf{v}\| = \sqrt{v_1^2 + \cdots + v_n^2}$$

To **normalize** the vector $\mathbf{v}$ consists in considering $\tilde{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$, i.e., the vector in the same direction as $\mathbf{v}$ that has unit length

A.3.4 Standard basis vectors

Vectors $\mathbf{i} = (1, 0, 0)$, $\mathbf{j} = (0, 1, 0)$ and $\mathbf{k} = (0, 0, 1)$ are the **standard basis vectors** of $\mathbb{R}^3$. A vector $\mathbf{v} = (v_1, v_2, v_3)$ can then be written as the linear combination of $\mathbf{i}, \mathbf{j}, \mathbf{k}$,

$$\mathbf{v} = v_1\mathbf{i} + v_2\mathbf{j} + v_3\mathbf{k}.$$

That representation is unique (in the basis $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$). For V $(\mathbb{R}^n)$, the standard basis vectors are usually denoted $\mathbf{e}_1, \ldots, \mathbf{e}_n$, with

$$\mathbf{e}_k = (\underbrace{0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{n-k+1}).$$

A.3.5 Dot product

Definition A.14 (Dot product)**.** *Let* $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$, $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$. *The* ***dot product*** *of* $\mathbf{a}$ *and* $\mathbf{b}$ *is the* ***scalar***

$$\mathbf{a} \bullet \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + \cdots + a_n b_n$$

The dot product is a special case of **inner product**.

Theorem A.15 (Properties of the dot product)**.** *For* $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^n$ *and* $\alpha \in \mathbb{R}$,

- $\mathbf{a} \bullet \mathbf{a} = \|\mathbf{a}\|^2$ *(so* $\mathbf{a} \bullet \mathbf{a} \geq 0$*, with* $\mathbf{a} \bullet \mathbf{a} = 0 \iff \mathbf{a} = \mathbf{0}$*)*
- $\mathbf{a} \bullet \mathbf{b} = \mathbf{b} \bullet \mathbf{a}$ *(*$\bullet$ *is commutative)*
- $\mathbf{a} \bullet (\mathbf{b} + \mathbf{c}) = \mathbf{a} \bullet \mathbf{b} + \mathbf{a} \bullet \mathbf{c}$ *(*$\bullet$ *distributive over* $+$*)*
- $(\alpha\mathbf{a}) \bullet \mathbf{b} = \alpha(\mathbf{a} \bullet \mathbf{b}) = \mathbf{a} \bullet (\alpha\mathbf{b})$
- $\mathbf{0} \bullet \mathbf{a} = 0$

A.3.6 Some results stemming from the dot product

Theorem A.16. *If* θ *is the angle between the vectors* $\mathbf{a}$ *and* $\mathbf{b}$*, then*

$$\mathbf{a} \bullet \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \, \cos\theta$$

Corollary A.17 (Cauchy-Schwarz inequality)**.** *For any two vectors* $\mathbf{a}$ *and* $\mathbf{b}$*, we have*

$$|\mathbf{a} \bullet \mathbf{b}| \leq \|\mathbf{a}\| \, \|\mathbf{b}\|$$

with equality if and only if $\mathbf{a}$ *is a scalar multiple of* $\mathbf{b}$*, or one of them is* $\mathbf{0}$*.*

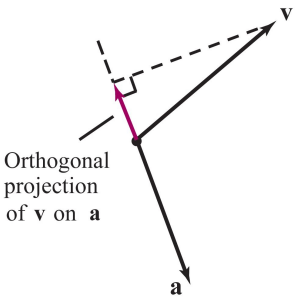
Theorem A.18. $\mathbf{a}$ *and* $\mathbf{b}$ *are orthogonal if and only if* $\mathbf{a} \bullet \mathbf{b} = 0$*.*

A.3.7 Scalar and vector projections

Let $\mathbf{a}, \mathbf{v}$ be vectors.

The **scalar projection** of $\mathbf{v}$ onto $\mathbf{a}$ (or **component** of $\mathbf{v}$ along $\mathbf{a}$) is given by

$$\text{comp}_{\mathbf{a}}\mathbf{v} = \frac{\mathbf{a} \bullet \mathbf{v}}{\|\mathbf{a}\|} \tag{A.1}$$



The **vector projection** (or **orthogonal projection**) of $\mathbf{v}$ onto $\mathbf{a}$ is given by

$$\text{proj}_{\mathbf{a}}\mathbf{v} = \left(\frac{\mathbf{a} \bullet \mathbf{v}}{\|\mathbf{a}\|}\right)\frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{\mathbf{a} \bullet \mathbf{v}}{\|\mathbf{a}\|^2}\mathbf{a} \tag{A.2}$$

A.4 Complex numbers

Definition A.19 (Complex numbers)**.** *A **complex number** is an ordered pair (a, b), where $a, b \in \mathbb{R}$. Usually written $a + ib$ or $a + bi$, where $i^2 = -1$ (i.e., $i = \sqrt{-1}$). The set of all complex numbers is denoted $\mathbb{C}$,*

$$\mathbb{C} = \{a + ib : a, b \in \mathbb{R}\}.$$

Definition A.20 (Addition and multiplication on $\mathbb{C}$)**.** *Letting $a + ib$ and $c + id \in \mathbb{C}$, addition on $\mathbb{C}$ is defined by*

$$(a + ib) + (c + id) = (a + c) + i(b + d)$$

and multiplication on $\mathbb{C}$ is defined by

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc).$$

Note that the latter is easy to obtain using regular multiplication and the fact that $i^2 = -1$.

Proposition A.21. *1. $\forall \alpha, \beta, \gamma \in \mathbb{C}$,*

2. $\alpha + \beta = \beta + \alpha$ and $\alpha\beta = \beta\alpha$ ***[commutativity]***

3. $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$ and $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ ***[associativity]***

4. $\gamma + 0 = \gamma$ and $\gamma 1 = \gamma$ ***[identities]***

5. $\forall \alpha \in \mathbb{C}$, $\exists \beta \in \mathbb{C}$ unique s.t. $\alpha + \beta = 0$ ***[additive inverse]***

6. $\forall \alpha \neq 0 \in \mathbb{C}$, $\exists \beta \in \mathbb{C}$ unique s.t. $\alpha\beta = 1$ ***[multiplicative inverse]***

7. $\gamma(\alpha + \beta) = \gamma\alpha + \gamma\beta$ ***[distributivity]***

Additive & multiplicative inverse, subtraction, division

Definition A.22. *Let $\alpha, \beta \in \mathbb{C}$*

- *$-\alpha$ is the **additive inverse** of α, i.e., the unique number in $\mathbb{C}$ s.t. $\alpha + (-\alpha) = 0$*
- ***Subtraction** on $\mathbb{C}$:*

$$\beta - \alpha = \beta + (-\alpha)$$

- *For $\alpha \neq 0$, $1/\alpha$ is the* ***multiplicative inverse*** *of α, i.e., the unique number in $\mathbb{C}$ s.t.*
$$\alpha(1/\alpha) = 1$$
- ***Division*** *on $\mathbb{C}$:*
$$\beta/\alpha = \beta(1/\alpha)$$

Definition A.23 (Real and imaginary parts)**.** *Let $z = a + ib$. Then $\Re(z) = a$ is* ***real part*** *and $\Im(z) = b$ is* ***imaginary part*** *of z.*

Definition A.24 (Conjugate and Modulus)**.** *Let $z = a + ib \in \mathbb{C}$. Then*
- *The* ***complex conjugate*** *of z is*
$$\bar{z} = a - ib.$$
- *The* ***modulus*** *(or* ***absolute value****) of z is*
$$|z| = \sqrt{a^2 + b^2} \geq 0.$$

Property A.25 (Properties of complex numbers)**.** *Let $w, z \in \mathbb{C}$, then*
- $z + \bar{z} = 2\Re z$
- $z - \bar{z} = 2i\Im z$
- $z\bar{z} = |z|^2$
- $\overline{w + z} = \bar{w} + \bar{z}$ *and* $\overline{wz} = \bar{w}\bar{z}$
- $\bar{\bar{z}} = z$
- $|\Re z| \leq |z|$ *and* $|\Im z| \leq |z|$
- $|\bar{z}| = |z|$
- $|wz| = |w|\,|z|$
- $|w + z| \leq |w| + |z|$ ***[triangle inequality]***

A.4.1 Solving quadratic equations

The interest of complex numbers is easily illustrated as follows. Consider the polynomial

$$P(x) = a_0 + a_1 x + a_2 x^2, \tag{A.3}$$

where $x, a_0, a_1, a_2 \in \mathbb{R}$. Letting

$$\Delta = a_1^2 - 4a_0 a_2,$$

we know from high school mathematics that if $\Delta > 0$, then

$$P(x) = 0$$

has two distinct *real* solutions,

$$x_1 = \frac{-a_1 - \sqrt{\Delta}}{2a_2} \quad \text{and} \quad x_2 = \frac{-a_1 + \sqrt{\Delta}}{2a_2},$$

while if $\Delta = 0$, then there is a (multiplicity 2) unique *real* solution

$$x_1 = \frac{-a_1}{2a_2}.$$

Finally, if $\Delta < 0$, then there is no (real) solution. Now suppose that instead of seeking real roots, we allow roots to be complex numbers. For this, consider again the polynomial (A.3). If instead of seeking $x \in \mathbb{R}$, we seek $x \in \mathbb{C}$, then the situation is the same, except when $\Delta < 0$. In the latter case, note that

$$\sqrt{\Delta} = \sqrt{(-1)(-\Delta)} = \sqrt{-1}\sqrt{-\Delta} = i\sqrt{-\Delta}.$$

As a consequence, when $\Delta < 0$, $-\Delta > 0$ and the square root is the usual one. To summarize, consider the polynomial (A.3). Letting

$$\Delta = a_1^2 - 4a_0a_2,$$

then

$$P(x) = 0$$

has two solutions,

$$x_{1,2} = \frac{-a_1 \pm \sqrt{\Delta}}{2a_2},$$

where, if $\Delta < 0$, $x_1, x_2 \in \mathbb{C}$ and take the form

$$x_{1,2} = \frac{-a_1 \pm i\sqrt{-\Delta}}{2a_2}.$$

A.4.2 Why this matters

Recall that the *eigenpairs* of the matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \tag{A.4}$$

are the λ and $\mathbf{v} \neq \mathbf{0}$ solutions to

$$A\mathbf{v} = \lambda\mathbf{v}.$$

We come back to this later in Section A.7.1, but let us recall how one treats this problem. Finding λ and $\mathbf{v} \neq \mathbf{0}$ such that $A\mathbf{v} = \lambda\mathbf{v}$ is equivalent, of course, to finding λ and $\mathbf{v} \neq 0$ such that

$$(A - \lambda\mathbb{I})\mathbf{v} = \mathbf{0}. \tag{A.5}$$

Since we seek nonzero $\mathbf{v}$, we must invoke the contraposite of Theorem A.62 and in particular, the following:

$$A\mathbf{v} = \mathbf{0} \text{ has infinitely many solutions } \iff \det(A) = 0.$$

(Note that the contrapositive of the statement "$A\mathbf{v} = \mathbf{0}$ has the unique solution $\mathbf{v} = \mathbf{0}$" in Theorem A.62 is "$A\mathbf{v} = \mathbf{0}$ has either no or infinitely many solutions". However, we know that $\mathbf{v} = \mathbf{0}$ is *always* solution to the homogeneous linear system $A\mathbf{v} = \mathbf{0}$ and as a consequence, the contraposite just states there are infinitely many solutions.)

As a consequence, we find the eigenvalues λ by solving $\det(A - \lambda\mathbb{I}) = 0$, i.e., finding the λ solutions to

$$|A - \lambda\mathbb{I}| = \begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix} = (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0.$$

Simplifying the latter polynomial,

$$\lambda^2 - (a_{11} + a_{22})\lambda + a_{11}a_{22} - a_{12}a_{21} = 0.$$

Let

$$P(\lambda) = \lambda^2 - (a_{11} + a_{22})\lambda + a_{11}a_{22} - a_{12}a_{21}$$

From previous discussion, letting

$$\begin{aligned} \Delta &= (a_{11} + a_{22})^2 - 4(a_{11}a_{22} - a_{12}a_{21}) \\ &= a_{11}^2 + a_{22}^2 + 2a_{11}a_{22} - 4a_{11}a_{22} + 4a_{12}a_{21} \\ &= a_{11}^2 + a_{22}^2 - 2a_{11}a_{22} + 4a_{12}a_{21} \\ &= (a_{11} - a_{22})^2 + 4a_{12}a_{21}, \end{aligned}$$

we have two (potentially equal) solutions to $P(\lambda) = 0$

$$x_{1,2} = \frac{a_{11} + a_{22} \pm \sqrt{\Delta}}{2}$$

that are complex if $\Delta < 0$.

Example: $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

A.5 Linear systems and matrices

A.5.1 Linear systems

Definition A.26 (Linear system)**.** *A **linear system** of m equations in n unknowns takes the form*

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_n \end{array} \tag{A.6}$$

The a_{ij}, x_j and b_j could be in $\mathbb{R}$ or $\mathbb{C}$, although here we typically assume they are in $\mathbb{R}$

The aim is to find $x_1, x_2, \ldots, x_n$ that satisfy all equations simultaneously

Theorem A.27 (Nature of solutions to a linear system)**.** *A linear system can have*

- *no solution*
- *a unique solution*
- *infinitely many solutions*

Operations on linear systems You learned to manipulate linear systems using

- Gaussian elimination
- Gauss-Jordan elimination

with the aim to put the system in **row echelon form** (REF) or **reduced row echelon form** (RREF)

Matrices and linear systems Writing

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

where A is an $m \times n$ **matrix**, $\mathbf{x}$ and $\mathbf{b}$ are n (column) **vectors** (or $n \times 1$ matrices), then the linear system in the previous slide takes the form

$$A\mathbf{x} = \mathbf{b}$$

Notation for vectors We usually assume vectors are column vectors and thus write, e.g.,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (x_1, x_2, \ldots, x_n)^T$$

Here, ^{T} is the **transpose operator** (more on this soon)

Consider the system

$$A\mathbf{x} = \mathbf{b}$$

If $\mathbf{b} = \mathbf{0}$, the system is **homogeneous** and always has the solution $\mathbf{x} = 0$ and so the "no solution" option in Theorem A.27 goes away

A.6 Matrix arithmetic

Definition A.28 (Matrix)**.** *An m-by-n or $m \times n$ matrix is a rectangular array of elements of $\mathbb{R}$ or $\mathbb{C}$ with m rows and n columns,*

$$A = [a_{ij}] = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

We always list indices as "row,column"

We denote $\mathcal{M}_{mn}(\mathbb{F})$ or $\mathbb{F}^{mn}$ the set of $m \times n$ matrices with entries in $\mathbb{F} = \{\mathbb{R}, \mathbb{C}\}$. Often, we omit $\mathbb{F}$ in $\mathcal{M}_{mn}$ if the nature of $\mathbb{F}$ is not important

When $m = n$, we usually write $\mathcal{M}_n$

Basic matrix arithmetic Let $A \in \mathcal{M}_{mn}, B \in \mathcal{M}_{mn}$ be matrices (of the same size) and $c \in \mathbb{F} = \{\mathbb{R}, \mathbb{C}\}$ be a scalar

- **Scalar multiplication**
$$cA = [ca_{ij}]$$
- **Addition**
$$A + B = [a_{ij} + b_{ij}]$$
- **Subtraction** (addition of $-B = (-1)B$ to A)
$$A - B = A + (-1)B = [a_{ij} + (-1)b_{ij}] = [a_{ij} - b_{ij}]$$
- **Transposition** of A gives a matrix $A^T = \mathcal{M}_{nm}$ with
$$A^T = [a_{ji}], \quad j = 1, \ldots, n, \quad i = 1, \ldots, m$$

Matrix multiplication The (matrix) **product** of A and B, AB, requires the "inner dimensions" to match, i.e., the number of columns in A must equal the number of rows in B

Suppose that is the case, i.e., let $A \in \mathcal{M}_{mn}$, $B \in \mathcal{M}_{np}$. Then the i, j entry in $C := AB$ takes the form
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Recall that the matrix product is not commutative, i.e., in general, $AB \neq BA$ (when both those products are defined, i.e., when $A, B \in \mathcal{M}_n$)

Special matrices

Definition A.29 (Zero and identity matrices)**.** *The **zero** matrix is the matrix 0_{mn} whose entries are all zero. The **identity** matrix is a square $n \times n$ matrix $\mathbb{I}_n$ with all entries on the main diagonal equal to one and all off diagonal entries equal to zero*

A.6.1 Symmetric matrices

Symmetric matrices occur quite frequently in this course, so let us go over some of their properties in some detail. First, recall that a symmetric matrix is defined as follows.

Definition A.30 (Symmetric matrix)**.** *A square matrix $A \in \mathcal{M}_n$ is **symmetric** if $\forall i, j = 1, \ldots, n$, $a_{ij} = a_{ji}$. In other words, $A \in \mathcal{M}_n$ is symmetric if $A = A^T$.*

Theorem A.31. *1. If $A \in \mathcal{M}_n$, then $A + A^T$ is symmetric.*

2. If $A \in \mathcal{M}_{mn}$, then $AA^T \in \mathcal{M}_m$ and $A^T A \in \mathcal{M}_n$ are symmetric.

Proof. 1. The statement is true if $A + A^T = (A + A^T)^T$. We have

$$(A + A^T)^T = A^T + (A^T)^T = A^T + A = A + A^T,$$

so the property is true.
2. AA^T is symmetric if $AA^T = (AA^T)^T$. We have

$$(AA^T)^T = (A^T)^T A^T = AA^T$$

and the property is true. Similarly, $(A^T A)^T = A^T (A^T)^T = A^T A$ and thus both AA^T and $A^T A$ are symmetric. □

Real symmetric matrices have a spectrum (set of eigenvalues) that has very specific properties.

Theorem A.32. *Let $A \in \mathcal{M}_n(\mathbb{R})$ be symmetric. Then all eigenvalues of A are real.*

Proof. Let $A \in \mathcal{M}_n(\mathbb{R})$ be symmetric and assume $(\lambda, \mathbf{v})$ is an eigenpair of A, i.e, $A\mathbf{v} = \lambda\mathbf{v}$ and $\mathbf{v} \neq \mathbf{0}$. Taking the complex conjugate, $\overline{A\mathbf{v}} = \overline{\lambda\mathbf{v}}$. Recall that $z \in \mathbb{C}$ is such that $z = \bar{z} \iff z \in \mathbb{R}$. So, since $A \in \mathcal{M}_n(\mathbb{R})$, $\overline{A} = A$ (consider the matrix entry by entry). Therefore,

$$A\bar{\mathbf{v}} = \overline{A}\bar{\mathbf{v}} = \overline{A\mathbf{v}} = \overline{\lambda\mathbf{v}} = \bar{\lambda}\bar{\mathbf{v}},$$

i.e., if $(\lambda, \mathbf{v})$ is an eigenpair, then $(\bar{\lambda}, \bar{\mathbf{v}})$ is also an eigenpair.

Still assuming that $A \in \mathcal{M}_n(\mathbb{R})$ is symmetric and $(\lambda, \mathbf{v})$ is an eigenpair of A, using what we just proved (that $(\bar{\lambda}, \bar{\mathbf{v}})$ also eigenpair), take transposes:

$$\begin{aligned} A\bar{\mathbf{v}} = \bar{\lambda}\bar{\mathbf{v}} &\iff (A\bar{\mathbf{v}})^T = (\bar{\lambda}\bar{\mathbf{v}})^T \\ &\iff \bar{\mathbf{v}}^T A^T = \bar{\lambda}\bar{\mathbf{v}}^T \\ &\iff \bar{\mathbf{v}}^T A = \bar{\lambda}\bar{\mathbf{v}}^T. \end{aligned}$$

Let us now compute $\lambda(\bar{\mathbf{v}} \bullet \mathbf{v})$. We have

$$\begin{aligned} \lambda(\bar{\mathbf{v}} \bullet \mathbf{v}) &= \lambda\bar{\mathbf{v}}^T\mathbf{v} = \bar{\mathbf{v}}^T(\lambda\mathbf{v}) \\ &= \bar{\mathbf{v}}^T(A\mathbf{v}) = (\bar{\mathbf{v}}^T A)\mathbf{v} \\ &= (\bar{\lambda}\bar{\mathbf{v}}^T)\mathbf{v} = \bar{\lambda}(\bar{\mathbf{v}} \bullet \mathbf{v}) \iff (\lambda - \bar{\lambda})(\bar{\mathbf{v}} \bullet \mathbf{v}) = 0. \end{aligned}$$

We have shown

$$(\lambda - \bar{\lambda})(\bar{\mathbf{v}} \bullet \mathbf{v}) = 0.$$

Let

$$\mathbf{v} = \begin{pmatrix} a_1 + ib_1 \\ \vdots \\ a_n + ib_n \end{pmatrix}.$$

Then

$$\bar{\mathbf{v}} = \begin{pmatrix} a_1 - ib_1 \\ \vdots \\ a_n - ib_n \end{pmatrix}.$$

So

$$\bar{\mathbf{v}} \bullet \mathbf{v} = (a_1^2 + b_1^2) + \cdots + (a_n^2 + b_n^2).$$

But $\mathbf{v}$, as an eigenvector, is $\neq \mathbf{0}$, so $\bar{\mathbf{v}} \bullet \mathbf{v} \neq 0$ and finally,

$$(\lambda - \bar{\lambda})(\bar{\mathbf{v}} \bullet \mathbf{v}) = 0 \iff \lambda - \bar{\lambda} = 0 \iff \lambda = \bar{\lambda} \iff \lambda \in \mathbb{R}.$$

The result is proved. □

By Theorem A.31, any matrix $A \in \mathcal{M}_{mn}$ gives rise to a symmetric matrix when it is multiplied to its transpose. A symmetric matrix generated this way enjoys even "better" spectrum properties.

Theorem A.33. *Let $A \in \mathcal{M}_{mn}(\mathbb{R})$. Then the eigenvalues of A^TA (and AA^T) are real* and nonnegative.

Proof. Let $A \in \mathcal{M}_{mn}(\mathbb{R})$. By Theorem A.31, A^TA is symmetric. It is also real, as the product of two real matrices. As a consequence, from Theorem A.32, A^TA has real eigenvalues.

Let then $(\lambda, \mathbf{v})$ be an eigenpair of A^TA, with $\mathbf{v}$ chosen so that $\|\mathbf{v}\| = 1$. Norms are functions $V \to \mathbb{R}_+$ (see Section A.3.3), so $\|A\mathbf{v}\|$ and $\|A\mathbf{v}\|^2$ are ≥ 0 and thus

$$\begin{aligned} 0 \leq \|A\mathbf{v}\|^2 &= (A\mathbf{v}) \bullet (A\mathbf{v}) = (A\mathbf{v})^T(A\mathbf{v}) \\ &= \mathbf{v}^T A^T A\mathbf{v} = \mathbf{v}^T(A^TA\mathbf{v}) = \mathbf{v}^T(\lambda\mathbf{v}) \\ &= \lambda(\mathbf{v}^T\mathbf{v}) = \lambda(\mathbf{v} \bullet \mathbf{v}) = \lambda\|\mathbf{v}\|^2 \\ &= \lambda, \end{aligned}$$

proving the result. □

A.6.2 Determinants

Definition A.34 (Determinant)**.** *Let $A \in \mathcal{M}_n$ with $n \geq 2$. The **determinant** of A is the* scalar

$$\det(A) = |A| = \sum_{j=1}^{n} a_{ij}C_{ij}$$

*where $C_{ij} = (-1)^{i+j}\det(A_{ij})$ is the (i,j)-**cofactor** of A and A_{ij} is the submatrix of A from which the ith row and jth column have been removed*

This is a cofactor expansion along the ith row

This is a recursive formula: it gives result in terms of n $\mathcal{M}_{n-1}$ matrices, to which it must in turn be applied, all the way down to

$$\det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Two special matrices and their determinants

Definition A.35. *$A \in \mathcal{M}_n$ is **upper triangular** if $a_{ij} = 0$ when $i > j$, **lower triangular** if $a_{ij} = 0$ when $j > i$, **triangular** if it is* either *upper or lower triangular and **diagonal** if it is* both *upper and lower triangular*

When A diagonal, we often write $A = \mathsf{diag}\ (a_{11}, a_{22}, \ldots, a_{nn})$

Theorem A.36. *Let $A \in \mathcal{M}_n$ be triangular or diagonal. Then*

$$\det(A) = \prod_{i=1}^{n} a_{ii} = a_{11}a_{22} \cdots a_{nn}$$

Inversion/Singularity

Definition A.37 (Matrix inverse)**.** *$A \in \mathcal{M}_n$ is **invertible** (or **nonsingular**) if $\exists A^{-1} \in \mathcal{M}_n$ s.t.*

$$AA^{-1} = A^{-1}A = \mathbb{I}$$

*A^{-1} is the **inverse** of A. If A^{-1} does not exist, A is **singular***

Theorem A.38. *Let $A \in \mathcal{M}_n$, $\mathbf{x}, \mathbf{b} \in \mathbb{F}^n$. Then*

- *A invertible $\iff$ $\det(A) \neq 0$*
- *If A invertible, A^{-1} is unique*
- *If A invertible, then $A\mathbf{x} = \mathbf{b}$ has the unique solution $\mathbf{x} = A^{-1}\mathbf{b}$*

Revisiting matrix arithmetic With addition, subtraction, scalar multiplication, multiplication, transposition and inversion, you can perform arithmetic on matrices essentially as on scalar, if you bear in mind a few rules

- The sizes have to be compatible
- The order is important since matrix multiplication is not commutative
- Transposition and inversion change the order of products:

$$(AB)^T = B^T A^T \text{ and } (AB)^{-1} = B^{-1}A^{-1}$$

A.7 Diagonalisation

A.7.1 Eigenvalues / Eigenvectors / Eigenpairs

Definition A.39. *Let $A \in \mathcal{M}_n$. A vector $\mathbf{x} \in \mathbb{F}^n$ such that $\mathbf{x} \neq \mathbf{0}$ is an **eigenvector** of A if $\exists \lambda \in \mathbb{F}$ called an **eigenvalue**, s.t.*

$$A\mathbf{x} = \lambda\mathbf{x}$$

*A couple $(\lambda, \mathbf{x})$ with $\mathbf{x} \neq \mathbf{0}$ s.t. $A\mathbf{x} = \lambda\mathbf{x}$ is an **eigenpair***

If $(\lambda, \mathbf{x})$ eigenpair, then for $c \neq 0$, $(\lambda, c\mathbf{x})$ also eigenpair since $A(c\mathbf{x}) = cA\mathbf{x} = c\lambda\mathbf{x}$ and dividing both sides by c..

Definition A.40 (Similarity)**.** *$A, B \in \mathcal{M}_n$ are **similar** ($A \sim B$) if $\exists P \in \mathcal{M}_n$ invertible s.t.*

$$P^{-1}AP = B$$

Theorem A.41 ($\sim$ is an equivalence relation)**.** *$A, B, C \in \mathcal{M}_n$, then*

- $A \sim A$ *($\sim$ **reflexive**)*
- $A \sim B \implies B \sim A$ *($\sim$ **symmetric**)*
- *$A \sim B$ and $B \sim C \implies A \sim C$* *($\sim$ **transitive**)*

Theorem A.42. *$A, B \in \mathcal{M}_n$ with $A \sim B$. Then*

- $\det\ A = \det\ B$
- *A invertible $\iff$ B invertible*
- *A and B have the same eigenvalues*

A.7.2 Diagonalisation

Definition A.43 (Diagonalisability)**.** *$A \in \mathcal{M}_n$ is **diagonalisable** if $\exists D \in \mathcal{M}_n$ diagonal s.t. $A \sim D$*

In other words, $A \in \mathcal{M}_n$ is diagonalisable if there exists a diagonal matrix $D \in \mathcal{M}_n$ and a nonsingular matrix $P \in \mathcal{M}_n$ s.t. $P^{-1}AP = D$

Could of course write $PAP^{-1} = D$ since P invertible, but $P^{-1}AP$ makes more sense for computations

Theorem A.44. *$A \in \mathcal{M}_n$ diagonalisable $\iff$ A has n linearly independent eigenvectors*

Corollary A.45 (Sufficient condition for diagonalisability)**.** *$A \in \mathcal{M}_n$ has all its eigenvalues distinct $\implies$ A diagonalisable*

For $P^{-1}AP = D$: in P, put the linearly independent eigenvectors as columns and in D, the corresponding eigenvalues

A.8 Linear independence/Bases/Dimension

Linear combination and span

Definition A.46 (Linear combination)**.** *Let V be a vector space. A **linear combination** of a set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ of vectors in V is a* vector

$$c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k$$

where $c_1, \ldots, c_k \in \mathbb{F}$

Definition A.47 (Span)**.** *The set of all linear combinations of a set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is the **span** of $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$,*

$$\mathsf{span}\ (\mathbf{v}_1, \ldots, \mathbf{v}_k) = \{c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k : c_1, \ldots, c_k \in \mathbb{F}\}$$

Finite/infinite-dimensional vector spaces

Theorem A.48. *The span of a set of vectors in V is the smallest subspace of V containing all the vectors in the set*

Definition A.49 (Set of vectors spanning a space)**.** *If* $\mathsf{span}\ (\mathbf{v}_1, \ldots, \mathbf{v}_k) = V$*, we say $\mathbf{v}_1, \ldots, \mathbf{v}_k$ **spans** V*

Definition A.50 (Dimension of a vector space)**.** *A vector space V is **finite-dimensional** if some set of vectors in it spans V. A vector space V is **infinite-dimensional** if it is not finite-dimensional*

Definition A.51 (Linear independence/Linear dependence)**.** *A set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ of vectors in a vector space V is **linearly independent** if*

$$(c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k = 0) \Leftrightarrow (c_1 = \cdots = c_k = 0)\,,$$

*where $c_1, \ldots, c_k \in \mathbb{F}$. A set of vectors is **linearly dependent** if it is not linearly independent.*

If linearly dependent, assume w.l.o.g. that $c_1 \neq 0$, then

$$\mathbf{v}_1 = -\frac{c_2}{c_1}\mathbf{v}_2 - \cdots - \frac{c_k}{c_1}\mathbf{v}_k$$

i.e., $\mathbf{v}_1$ is a linear combination of the other vectors in the set.

Theorem A.52. *Let V be a finite-dimensional vector space. Then the **cardinal** (number of elements) of every linearly independent set of vectors is less than or equal to the number of elements in every spanning set of vectors.*

E.g., in $\mathbb{R}^3$, a set with 4 or more vectors is automatically linearly dependent.

Definition A.53 (Basis)**.** *Let V be a vector space. A **basis** of V is a set of vectors in V that is both linearly independent and spanning.*

Theorem A.54 (Criterion for a basis)**.** *A set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ of vectors in a vector space V is a basis of $V \iff \forall \mathbf{v} \in V$, $\mathbf{v}$ can be written uniquely in the form*

$$\mathbf{v} = c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k,$$

where $c_1, \ldots, c_k \in \mathbb{F}$.

Plus/Minus Theorem

Theorem A.55 (Plus/Minus Theorem)**.** *S a nonempty set of vectors in vector space V*

- *If S is linearly independent and $V \ni \mathbf{v} \notin \mathsf{span}\ (S)$, then $S \cup \{\mathbf{v}\}$ is linearly independent*
- *If $\mathbf{v} \in S$ is linear combination of other vectors in S, then $\mathsf{span}\ (S) = \mathsf{span}\ (S - \{\mathbf{v}\})$*

More on bases

Theorem A.56 (Basis of finite-dimensional vector space)**.** *Every finite-dimensional vector space has a basis*

Theorem A.57. *Any two bases of a finite-dimensional vector space have the same number of vectors*

Definition A.58 (Dimension)**.** *The **dimension** $\dim V$ of a finite-dimensional vector space V is the number of vectors in any basis of the vector space*

Theorem A.59 (Dimension of a subspace)**.** *Let V be a finite-dimensional vector space and $U \subset V$ be a subspace of V. Then $\dim U \leq \dim V$*

Constructing bases

Theorem A.60. *Let V be a finite-dimensional vector space. Then every linearly independent set of vectors in V with $\dim V$ elements is a basis of V*

Theorem A.61. *Let V be a finite-dimensional vector space. Then every spanning set of vectors in V with $\dim V$ elements is a basis of V*

A.9 Linear algebra in a nutshell

To finish, here is the "expanding result" that you kept adding to in your first Linear Algebra course. Of course, it can still be expanded... As a "TFAE statement", either all statements in the list are true simultaneously, or (exclusively) they are all false.

Theorem A.62. *Let $A \in \mathcal{M}_n$. The following statements are equivalent (TFAE):*

1. *The matrix A is invertible (or nonsingular).*
2. *$\forall \mathbf{b} \in \mathbb{F}^n$, $A\mathbf{x} = \mathbf{b}$ has a unique solution ($\mathbf{x} = A^{-1}\mathbf{b}$).*
3. *The only solution to $A\mathbf{x} = \mathbf{0}$ is the trivial solution $\mathbf{x} = \mathbf{0}$.*
4. *$RREF(A) = \mathbb{I}_n$.*
5. *The matrix A is equal to a product of elementary matrices.*
6. *$\forall \mathbf{b} \in \mathbb{F}^n$, $A\mathbf{x} = \mathbf{b}$ has a solution.*
7. *There is a matrix $B \in \mathcal{M}_n$ such that $AB = \mathbb{I}_n$.*
8. *There is an invertible matrix $B \in \mathcal{M}_n$ such that $AB = \mathbb{I}_n$.*
9. *$\det(A) \neq 0$.*
10. *0 is not an eigenvalue of A.*

Index

n-clique, 71

adjacent vertices, 68
antipodal vertices, 112
arcs, 88
articulation point, 80
articulation set, 94

basis, 148
best approximation, 30
binary relation, 65
bipartite graph, 71
boundary of a region in a plane graph, 82
branches, 94
bridge, 80

cardinal, 148
chromatic number, 83
circuit, 75
closed walk, 75
complete bipartite graph, 71
complex conjugate, 139
component, 38
connected components, 79
connected graph, 78
connected vertices, 78
cut vertex, 80
cycle, 75

data wrangling, 11
degree matrix, 103
degree, digraph, 90
diagonalisable, 147
digraph, 88
directed cycle, 91
directed path, 91
directed trail, 91
directed walk, 91
disconnected digraph, 93
disconnected graph, 78
dot product, 137

edges, 66
elements, 133
equivalence classes, 65
error vector, 26
even vertex, 69

finite graph, 68
forest, 94

genetic algorithm, 27
geodesic distance, 108

Hamiltonian cycle, 78
Hamiltonian graph, 78
Hamiltonian path, 77

indegree, 90
initial endpoint, 89
intersection, 134
irreducible matrix, 104

Laplacian matrix, 103
least squares solution, 29
length of a cycle, 75
linear combination, 148
linearly dependent, 148
linearly independent, 148

minimisation problem, 27
modulus, 139
multidigraph, 88
Multiple arcs, 88

neighbours, 89
nonplanar graph, 82
norm, 136
normal equations, 30

odd vertex, 69
order of a graph, 67
orientable graph, 94
orienting the graph, 94
orthogonal basis, 36
orthogonal matrix, 37
orthogonal projection, 38, 138
orthogonal set, 35
orthonormal basis, 36
orthonormal set, 36
outdegree, 90

path, 75
pendant vertex, 96
planar graph, 82
predecessor, 89

quantifiers, 134

reducible matrix, 104
reflexive relation, 65
region in a plane graph, 82
right singular vectors, 40

scalar projection, 138
set, 133
singular value decomposition, 40
singular values, 39
sink, 89
size of a graph, 68
source, 89
span, 148
spectrum of a graph, 103
strong orientation, 94
strongly connected, 93
strongly connected components, 93
subset, 133
successor, 89
symmetric matrix, 143

terminal endpoint, 89
traceable graph, 78
trail, 75
traversable graph, 76
tree, 94

underlying graph, 92
undirected graph, 66
union, 134

vertices, 66

walk, graph, 75
weakly connected, 92