Contents

Introduction

1.1 How it's different from Word

1.1.1 WYSIWYG - What You See Is What You Get

You may be familiar with the challenge that is using Word to write equations even as simple as those in Figure 1.1. The formatted text on the page is exactly everything you have, which often means spending time trying to fix how it looks and what goes where, a process we call typesetting. So **what is the alternative?**

The Laplace transform of a function $f(t)$, defined for all real numbers $t \geq 0$, is the function $F(s)$, which is a unilateral transform defined by:

$$F(s) = \int_0^\infty f(t)e^{-st}dt$$

Figure 1.1: Word, an example of What You See Is What You Get

1.1.2 Typesetting

Let me introduce you to LaTeX, pronounced either *lah-tec* or *lay-tec*.

– **stuff here**.

The main motivators for why you would want to use it are:

1. Beautifully written documents. No more trying to deal with the various fonts, margins, spacing, etc.;
2. Easy bibliography management, citations and cross-references;
3. Easy equations and graphs. Even extremely complicated mathematical concepts can be easily written.

Let's go back to our example, and see how it would be done with LaTeX:

Listing 1.1: Example document written with LaTeX

```
1 \documentclass[12pt,a4paper]{article}
2 \usepackage{amsmath}
3 \begin{document}
4 The Laplace transform of a function $f(t)$, defined for all real numbers $t \geq 0$ is the
      function $F(s)$, which is a unilateral transform defined by:
5 \begin{equation*}
6     F(s) = \int_0^\infty f(t)e^{-st} dt
7 \end{equation*}
8 \end{document}
```

If you are familiar with programming, this may vaguely look like a *markup language*, and it is! If you are not, don't worry. We will introduce the most distinctive features, and go into more detail as necessary.

1.1.3 Markup language

There are generally three types of markups:

1. **Environments** introduce some kind of formatting, such as lists, math mode or more complicated things. A backslash, `\`, is used to indicate a markup, curly brackets `{}` indicate an *argument*, and square brackets `[]` (optionally) provide options.

   With very few exceptions, environments have the following format:

```
\begin{environment-name}[options]
    ...
\end{environment-name}
```

   One of the exceptions is in Listing 1.1. Can you spot it? It is `$...$`, the *in-line math mode* environment. We will be exploring it in more detail later.

2. **Instructions** define some feature of the document. This ranges from breaking a page to defining the headings you have available. In our example you can see:

   `\documentclass[...]{article}`

3. **Variables** can either be predefined or user defined, and these range from greek letters to the integral sign to a whole expression. Some examples seen are `\geq` (**g**reater or **eq**ual) and `\int` ($\int$). Intuitively, `\alpha` results in α, and similarly for other greek letters.

   Note: There are some characters with predefined meaning, such as `{`, `}` and `&`. To use the literal curly brackets, ampersand, etc we would need to *escape* them. Conveniently, this is done with a backslash (\), so feel free to think of it as a variable: `\{`, `\}` and `\&`.

1.1.4 Packages

You may have noticed that `\usepackage{amsmath}` was not mentioned as an instruction the previous section. That's because packages are worth mentioning on their own.

Packages add features to our document, similar to *import* in most programming languages. `amsmath` gives us a wide array of maths tools, but there are packages for draw-

ing, graphing, colouring, better management of bibliography, easier organisation of your files... and the list goes on.

1.1.5 Compilation

We can use any text editing software to write and save our documents in `.tex` files. In order to produce a `.pdf`, it needs to be *compiled*. As a beginner you may spend a lot of time scratching your head, wondering why it's not compiling.

The good news is that recommended IDEs (**I**ntegrated **D**evelopment **E**nvironments, basically text editors filled with features) handle the compilation for you and have features to help you spot mistakes.

Installation

2.1 Installing LaTeX

2.2 Installing an IDE

Getting started

3.1 My first document

A LaTeX file has a `.tex` extension, and begins with what we call a *preamble* — declaring the *class* of document, followed by `\begin{document}...\end{document}`.

```
1 \documentclass[12pt]{article}
2 \begin{document}
3 This is the first sentence of the first paragraph. Second sentence of first paragraph.
4 Third sentence first paragraph.
5
6 This is the second paragraph
7 \end{document}
```

Producing the following output:

This is the first sentence of the first paragraph. Second sentence of first paragraph. Third sentence first paragraph.

This is the second paragraph

Generally, we declare a document class with `\documentclass[option1, ...]{class}`, with the most commonly used classes being `article` and `report`. Every class has a different set of default behaviours, such as `report` providing a title page, but we can give give it *options*. The option we set was to change the font size from the default `10pt` to `12pt`.

Another option commonly used in academia is `twocolumn` to produce a two column document. You can find more options and information on default behaviour on this link.

Later on we will come back to the *preamble* for other important commands. Generally we create templates, so it isn't necessary to remember every small detail, and very quick to get started on a new document.

3.1.1 Paragraph

You will notice that the first paragraph consists of both lines 3 and 4. This is because a paragraph is only created by having a full empty line (like line 5). One advantage to separating sentences by a new line is that you can more readily move, copy and delete them in your editor.

Another important feature is that indentation was made automatically. LaTeXis smart enough to indent for you and almost always get it right. If you really want to force a paragraph without indentation, use \\ at the end of the previous one, like so:

```
1    paragraph one\\
2    paragraph two not indented.
```

Note Including an empty line after \\ will result in a very common warning: `Underfull hbox`. More information on this later in the common errors and warnings section.

3.1.2 Sectioning

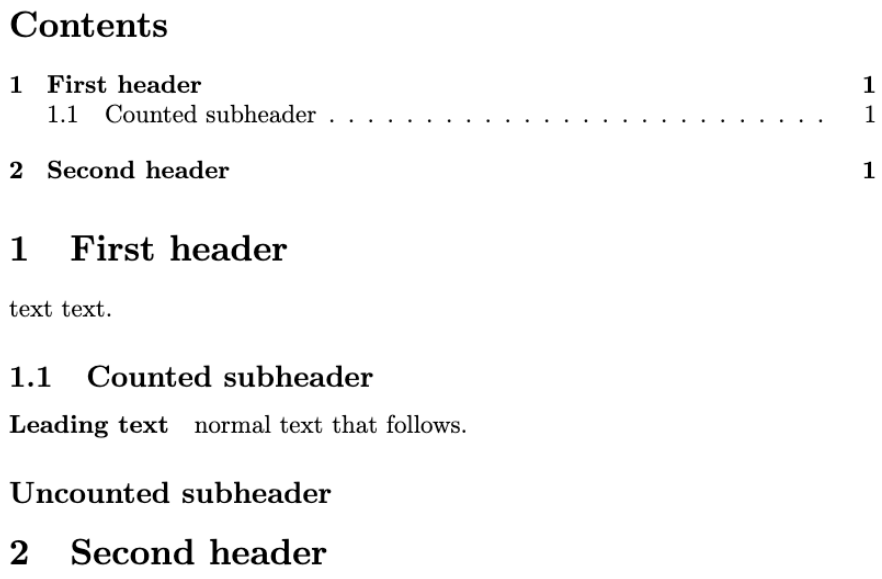
The basic way we separate documents is into `section`, `subsection` and `paragraph`.

Listing 3.1: Caption

```
1  \documentclass{article}
2  \begin{document}
3  \tableofcontents
4  \section{First header}
5  text text.
6  \subsection{Counted subheader}
```

```
\paragraph{Leading text}
normal text that follows.
\subsection*{Uncounted subheader}
\section{Second header}
\end{document}
```

Results in:



Contents

1 First header 1
1.1 Counted subheader 1

2 Second header 1

1 First header

text text.

1.1 Counted subheader

Leading text normal text that follows.

Uncounted subheader

2 Second header

Every tag that includes some form of counting can have an asterisk (*) to remove the counting. In this case, you can see the difference between `\section{}` and `\section*{}`, and most importantly, the table of contents, generated with `\tableofcontents`, excluded the uncounted subheader.

The `report` class also offers `\chapter(*){}` and `\part(*){}`, relevant mostly to very large documents.

Note: You will notice that the table of content and the actual content are in the same page. If you want a page break at any point, just use `\pagebreak`!

3.1.3 Bold, italic, underline, etc

```
1 \textit{Italics}, \underline{underline}, \textbf{bold}.
2 \textit{Emphasis switches from italics to \emph{normal}} and \emph{vice-versa} based on
      context.
3 \texttt{And we can even get monospace!}
```

Results in:

Italics, underline, **bold**. *Emphasis switches from italics to* normal and *vice-versa* based on context. `And we can even get monospace!`

VSCode has a shortcut for these, so you don't need to remember the exact keyword. `Ctrl+L` to initiate a LaTeX command, then `Ctrl+` the first letter of the command — `Ctrl+i`, `Ctrl+b`, `Ctrl+u`, `Ctrl+e` or `Ctrl+t`, respectively. So for bold, you would press is `Ctrl+L+Ctrl+B`. For Mac, just replace `Ctrl` for `Cmd`.

Note: Generally the suggestion is to use `\emph{}` over `\textit{}`. Think of it as a "generic highlighter" that you can modify with default behaviour to *italicise*, but you could make it change colours or font size or anything else.

3.2 Bibliography management

The tool that allows us to easily manage bibliography is called BiBTeX. In particular, we are using a *package* called `natbib` that gives us some extra features. Using it has two distinct moments: Adding an entry to our bibliography, and citing.

3.2.1 Creating a bibiliography

A bibliography file has a `.bib` extension, and each entry has a very specific format. Let's start with creating the file `bibliography.bib`, so our working directory looks like this:

```
Example
├── bibliography.bib
└── example.tex
```

Each entry has a source, whether `article`, `book`, `misc` or many more and has the following format:

```
@article{ GerberLeahR2005, %Unique identifier
  author    = {Gerber, Leah R and Beger, Maria and McCarthy, Michael A and Possingham, Hugh P},
  title     = {A theory for optimal monitoring of marine reserves},
  issn      = {1461-023X},
  journal   = {Ecology letters},
  pages     = {829--837},
  volume    = {8},
  publisher = {Blackwell Science Ltd},
  number    = {8},
  year      = {2005},
  edition   = {Editor, Ransom Myers Manuscript received 15 March 2005 First decision made 21 April 2005 Manuscript accepted 6 May 2005},
},
```

It's worth highlighting that the basic format is essentially `@article{ID,...}`, with each entry being separated by commas. BiBTeX will handle "et al" and other conventions as long as you stick to the following format:

```
author = {LastName1, FirstName1 and LastName2, FirstName2 and...}
```

The good side is that you rarely, if ever, need to type it out yourself. When you find an article through UCL's library, JAMA, Science Direct and many other resources, there will be an option to **export citation to BiBTeX**. Simply copy the contents to your bibliography file and you're ready to cite!

3.2.2 Manually adding a bibliography entry

While using VSCode, you can scaffold a bibliography entry in a `.bib` file. Simply type `@` and press `Ctrl+space`, and it will generate the skeleton for an entry. Generally this is only used for citing random websites, so you will want to pick the `@misc` option.

3.2.3 Citations

Now we just need to let our document know where to find our bibliography file and we can use `\cite{}` to include citations. Every entry that that is cited, automatically gets added to a Reference section at the end of your document.

Listing 3.2: example.tex

```
1 \documentclass[]{article}
2 \usepackage[square,numbers]{natbib}
3 \bibliographystyle{unsrtnat}
4 \begin{document}
5 Citations are made so easy with \LaTeX, can you see \cite{GerberLeahR2005}?
6 \bibliography{bibliography}
7 \end{document}
```

Giving us the following output:

Citations are made so easy with LaTeX, can you see [1]?

References

[1] Leah R Gerber, Maria Beger, Michael A McCarthy, and Hugh P Possingham. A theory for optimal monitoring of marine reserves. *Ecology letters*, 8(8): 829–837, 2005. ISSN 1461-023X.

`\usepackage[square,numbers]{natbib}` is what determines that in-text citations is **[1]**. Alternatively if you prefer **(Gerber et al, 2005)**, use `\usepackage[round]{natbib}`, and `\citep{}` instead of `\cite{}`:

```
\usepackage[round]{natbib}
...
\citep{GerberLeahR2005}
```

The `natbib` package gives us the `bibliographystyle{unsrtnat}` option, which determines the style of the references. There are others, as well as more information on `natbib`, which you can find more about on this link.

A really important feature is that the editor even suggests the authors we have added to our `.bib` file. Automatically generating numbers for figures, tables and correcting any citations is a key feature of LaTeX. This means we can easily refer to a figure, move it around and it will correctly choose its number.

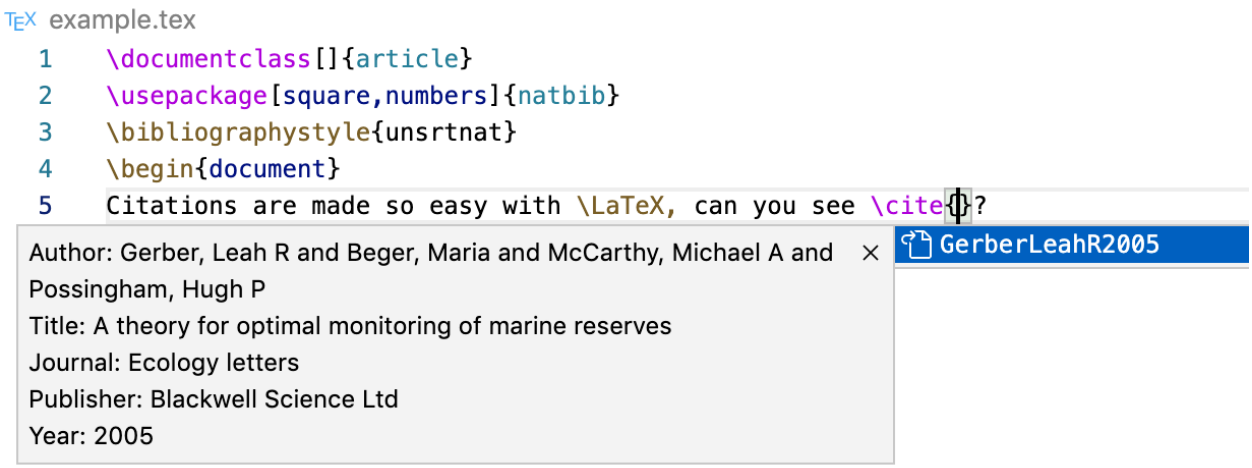


Figure 3.1: Autocomplete from our bibliography file

Before we get into adding figures, tables, cross-referencing, etc, it is a great idea to take a detour and discuss organisation.

3.3 Staying organised

So far our examples have been extremely short, but imagine having dozens of chapters with dozens of packages and whatever configuration is necessary for them. One thing we can do is split

3.4 Maths

3.4.1 Equations, sums and alignment

3.4.2 Use of variables

3.4.3 Vectors and Matrices

3.4.4 Calculus

3.5 Graphs with Tikz

3.6 Circuits

3.7 Control systems

How do I...?

4.1 Search engine

4.2 Stack Exchange

4.3 CTAN package information

Working faster

5.1 Becoming familiar with the IDE

5.2 Using snippets