

@ \ \ , \ ; \ : \ ! \ - \ = \ > \ < \ + \ ' \ ` \ | \ (\) \ [\] \ addcontentsline
\ addtocontents \ addtocounter \ address \ addtolength \ addvspace \ alph \ ap
\ arabic \ author \ backslash \ baselineskip \ baselinestretch \ begin \ bfser
\ bibitem \ bigskipamount \ bigskip \ boldmath \ boldsymbol \ cal \ caption \ c
\ centering \ chapter \ circle \ cite \ clearatdoublepage \ clearpage \ cline \ c
\ color \ copyright \ dashbox \ date \ ddots \ documentclass \ dotfill \ em \ em
\ end \ ensuremath \ epigraph \ euro \ fbox \ flushbottom \ fnsymbol \ footnote
\ footnotemark \ footnotesize \ footnotetext \ frac \ frame \ framebox
\ frenchspacing \ hfill \ hline \ href \ hrulefill \ hspace \ huge \ Huge
\ hyphenation \ include \ includegraphics \ includeonly \ indent \ input \ its
\ item \ kill \ label \ large \ Large \ LARGE \LaTeX \LaTeXe \ldots \left \le
\ line \ linebreak \ linethickness \ linewidth \ listoffigures \ listoftables
\ location \ makebox \ maketitle \ markboth \ markright \ mathcal \ mathop \ mbo
\ medskip \ multicolumn \ multipt \ newcommand \ newcolumntype \ newcounter
\ newenvironment \ newfont \ newlength \ newline \ newpage \ newsavebox \ newtl
\ nocite \ noindent \ nolinebreak \ nonfrenchspacing \ normalsize \ nopagebre
\ not \ onecolumn \ opening \ oval \ overbrace \ overline \ pagebreak \ pagenum
\ pageref \ pagestyle \ par \ paragraph \ parbox \ parindent \ parskip \ part
\ protect \ providecommand \ put \ quad \ qqquad \ raggedbottom \ raggedleft
\ raggedright \ raisebox \ ref \ renewcommand \ right \ rmfamily \ roman \ rule
\ savebox \ sbbox \ scshape \ scriptsize \ section \ setcounter \ setlength
\ settowidth \ sffamily \ shortstack \ signature \ slshape \ slash \ small
\ smallskip \ sout \ space \ sqrt \ stackrel \ stepcounter \ subparagraph
\ subsection \ subsubsection \ tableofcontents \ telephone \TeX \textbf
\ textcolor \ textit \ textmd \ textnormal \ textrm \ textsc \ textsf \ textsl
\ texttt \ textup \ textwidth \ textheight \ thanks \ thispagestyle \ tiny \ ti
\ today \ ttfamily \ twocolumn \ typeout \ typein \ ueline \ underbrace \ underl
\ unitlength \ usebox \ usecounter \ uwave \ value \ vbox \ vcenter \ vdots \ ve
\ verb \ vfill \ vline \ vphantom \ vspace

Writing science with \LaTeX

Peter Jansson

Contents

1	Scientific writing and L^AT_EX	6
2	A quick start to writing with L^AT_EX	7
3	Expanding your manuscript	12
3.1	Types of commands	12
3.2	The document structure	13
3.3	Document classes	14
3.4	Reserved characters	14
3.5	Type face styles	15
3.6	Typographic features	16
3.7	Breaks and space	17
3.8	Headings	18
3.9	Table of contents	19
3.10	Index	20
3.11	Environments	21
3.12	cross-referencing	22
3.13	Special commands	23
4	Tabular information	24
5	Graphics	26
6	Floats; tables and figures on the run	27
7	Mathematical typesetting	28
8	Extending the functionality of L^AT_EX with packages	30
8.1	amslatex	31
8.2	appendix	31
8.3	babel	32
8.4	booktabs	32
8.5	caption	33
8.6	dcolumn	33
8.7	float	34
8.8	fontspec	34
8.9	geometry	35
8.10	graphicx	35
8.11	hyperref	36
8.12	inputenc	37
8.13	lineno	37
8.14	lipsum	37
8.15	multirow	38
8.16	natbib	38
8.17	parskip	40
8.18	siunitx	40
8.19	threeparttable	41
8.20	tikz/pgf	42
8.21	titlesec	45
8.22	xcolor	45
9	References	46
9.1	Using a reference manager	46
9.2	Integration of Overleaf and Zotero	47
9.3	The natbib package	48
9.4	The fully automated reference system	49
9.5	The semi-automated reference system	50

10 The role of type setting	51
10.1 Page formatting	52
10.2 Placement of figures and tables	53
10.3 Final editing of your document	53
11 Automating L^AT_EX or creating your own commands	55
12 Copying documents from Word to L^AT_EX	58
13 Making presentation slides in L^AT_EX	59
14 Reinventing the wheel? ... or not	60
15 How does it really work?	61
16 Saving the worst to last	63
Acknowledgement	64
Appendices	65
A Running L^AT_EX on your own computer	65
A.1 L ^A T _E X	65
A.2 L ^A T _E X editors	65
A.3 Reference management	66
A.4 Yet more software that might be of use	66
B Install L^AT_EX on your computer	66
B.1 Windows	66
B.2 Mac	67
B.3 Linux	67
C Learning more on L^AT_EX	67
References	69
Index	71



Scientific writing with L^AT_EX is copyright © 2023 by Peter Jansson and available through its doi: <https://dx.doi.org/10.17045/sthlmuni.3205753>
This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Preface

This document is intended to provide you with a good working knowledge of \LaTeX so that you can freely use it for your scientific output. \LaTeX was, and is, created by scientists for scientists and is widely used by publishers for both journals and books. The origin of \LaTeX is \TeX , or more precisely $\tau\epsilon\chi$ (pronounced te:ch), the basic engine created by Donald Knuth and released in 1978 (Knuth, 1984) to enable mathematicians to generate proper printouts of mathematical notation. Knuth, however, went further and made sure \TeX included proper typographical capabilities so as to generate typographically accurate documents. Leslie Lamport developed a more descriptive mark-up language to access the \TeX engine in 1984 (Lamport, 1994) which was named \LaTeX (pronounced late:ch or laytech¹) as an abbreviation of *Lamport* \TeX . \LaTeX has been adopted by researchers and publishers as a tool for generating complex scientific texts. To have working knowledge of \LaTeX should therefore be part of every scientist's toolbox.

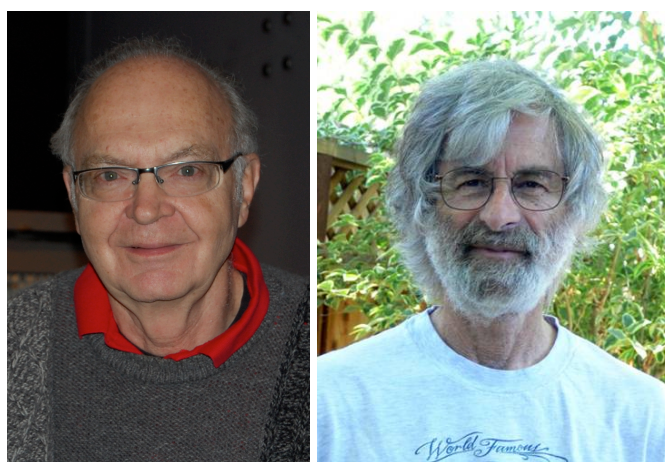


Figure 1. Donald Knuth (left) created the type setting system \TeX in 1978 (Knuth, 1984) which was developed further by Leslie Lamport (right) in 1984 into \LaTeX (Lamport, 1994, Lamport \TeX). (Source: Wikimedia Commons: Donald Knuth and Leslie Lamport)

There are two choices for running \LaTeX , on-line or off-line tools. These should not be considered mutually exclusive, you may need to work both ways. In this text I will assume you are working with the on-line platform [Overleaf](#) which is what I would also recommend to anyone, beginner or experienced, since all you need is a web-browser and an internet connection, no installation on your computer. Overleaf is similar to Google Docs in that documents are stored in the cloud and that several persons can interactively collaborate on a document. The drawback is of course that you cannot work on your documents without internet access. You can, however, synchronise your files in the cloud with [Dropbox](#) or [Git](#) so that you always have updated versions on your own computer. To work off-line you need to install a \LaTeX environment locally on your own computer; this is covered in appendix A.

So I strongly advice the beginner to sign up for and use [Overleaf](#). Note that you may be covered by an institutional license if you study or work at a university. Please visit the Overleaf site and look at the introductory video to familiarise yourself with the Overleaf environment. This document was entirely written on and using Overleaf but does not detail how you use the interface. Of particular use is the [keyboard shortcuts](#) sheet.

Although not covered in any detail in this document, I also strongly urge you to start using a reference manager for handling your scientific references. Unless you already use a manager

¹Under no circumstances should the word \LaTeX be literally pronounced as latex, the natural emulsion of polymer micro-particles or non-vulcanised rubber material.

I would recommend using [Zotero](#) which integrates with Overleaf. This integration is briefly covered in section 9.2.

In this text, I will use typewriter type face to highlight L^AT_EX code. In the cases where a complete section of code is given

```
1 it will be presented in a blue field.
```

Apart from what is covered in this text, you probably should download the [L^AT_EX 2_ε Cheat Sheet](#) and the [L^AT_EX quick reference](#) and keep them handy. The [comprehensive list of L^AT_EX symbols](#) is also useful to have handy. You should also be aware of the [T_EX StackExchange](#) site, which is a question and answer site where probably all your questions are already answered. At the time of writing there are 253 673 answered questions on the site. You can also find many solutions to type-setting issues by typing ‘latex’ followed by whatever you are looking for. It is almost certain some of the highest hits are from the T_EX StackExchange.

More details on how different macro packages work can be obtained from their documentation which is available on the [Comprehensive T_EX Archive Network \(CTAN\)](#) site. At the time of writing there are 6482 macro packages contributed by 2946 different users. Of course, far from all will be useful to any one user.

1 Scientific writing and \LaTeX

Apart from the linguistic aspects of scientific writing, such as clarity, brevity and density, what distinguishes scientific writing is the use of many technical terms, technical notation and cross-referencing of citations, figures, tables and equations as well as the need to follow precise document formatting instructions. Clearly you can accomplish the first aspect with any type of writing tool from the Goose pen to a modern word processor, it is the second part that requires some special tools. This is where \LaTeX fits.

First, I need to explain what \LaTeX is and what it is not. A word processor such as Word is a tool for writing text and creating documents in a linear fashion. You enter text and you see the result immediately but you may realise what happens with other pages in the document, particularly with figures and tables. \LaTeX is not a word processor, it is *a system* that takes a text file containing text and instructions for how the text should be formatted and compiles the file to generate a document. In this sense it is akin to manually creating an html-page or writing a script for computing. In fact learning \LaTeX is probably easier to those who have some prior programming experience, but that should not dismay anyone from giving it a try. In processing a document \LaTeX takes the instructions and optimises the entire document to look as good as possible in all its parts. This means \LaTeX considers what happens both before and after the text you type in to make sure the entire text looks good.

Now \LaTeX is not the universal response to all writing issues; as with everything there are definite pro's and con's. Since \LaTeX involves knowing, or at least have familiarity with, a number of commands and understanding how to use them, it is associated with a learning curve similar to that of learning a programming language. In fact, it is a programming language (Turing complete²). It is therefore evident that it makes little sense to learn \LaTeX for menial tasks such as writing a letter or a memo to colleagues or a list for making purchases, that is the realm of Word. \LaTeX comes into its own when the document is more complex such as a thesis or scientific paper.

There are probably many different imaginable tasks where \LaTeX is preferable, I will just mention a few. First, it may be useful to say that what tasks where you decide to use \LaTeX will likely depend on how well you know \LaTeX . I use \LaTeX for most of my writing but then I have spent time learning and getting acquainted with \LaTeX . But, I am still not an expert on all things \LaTeX !

When you write a scientific article you greatly benefit from using \LaTeX because you can clearly express, for example, complicated notation and mathematics. It is also very likely that the journal accepts \LaTeX manuscripts in their own journal format. \LaTeX is also very useful when you try to write a long document, book or similar, where you need cross referencing, you may need to create a table of contents and perhaps an index; tools built into \LaTeX . A third type of document is that which has to be formatted exactly the same time after time. This could, for example, be a report series from a project or organisation or from ongoing research. In all such instances, there is little competition.

So the conclusion is that the more complex your writing task the more you will benefit from using \LaTeX . Another way to describe this is that what is difficult in Word is simple in \LaTeX and *vice versa*. Figure 2 shows a qualitative comparison between Word and \LaTeX indicating where each is advantageous to use. Word is clearly requires less effort to produce simple documents whereas \LaTeX due to its learning curve requires more to accomplish the same simple documents. Since a natural science document most often require quite technical notation it will likely reach higher complexity sooner than a document in the humanities. In both cases Word will tend towards the impossible quite quickly while \LaTeX will allow you to make very complex

²If a programming language allow sequences of code, including logical if-then-else statements and some form of iteration such as while or for loops, it is *Turing complete*.

documents with less increase in effort in the long-term. This has led to the generalisation that what is easy in Word is complex in \LaTeX and *vice versa*. So it is clear that you need to choose wisely depending on what kind of document you intend to produce.

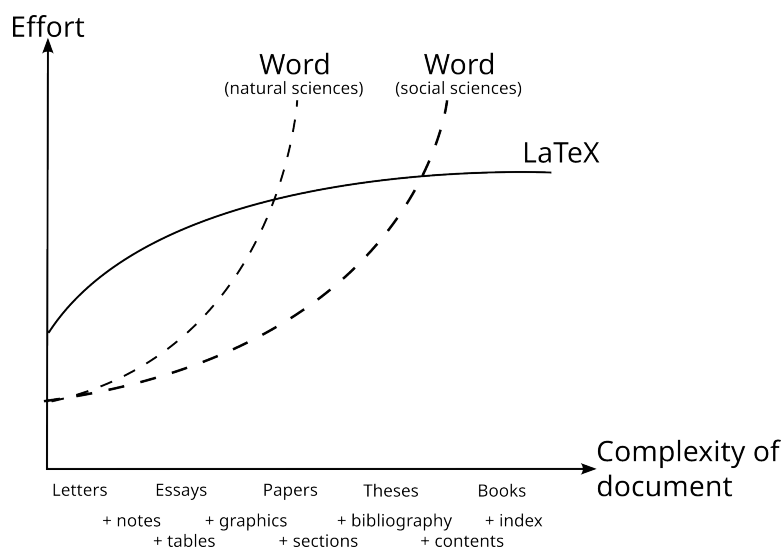


Figure 2. Conceptual figure showing a comparison of the capability of Word and \LaTeX in terms of the effort required to accomplish different complexity documents. After original figure from [Clemens Lode](#).

Many seem to think that Word *versus* \LaTeX is something similar to supporting, for example, different football clubs but that is just plain stupid or unimaginably ignorant. You should choose the tool that suits your immediate needs the best. At the same time you should not avoid a tool just because it may require you to spend some time learning it. You need to expand your academic toolbox in order to be as efficient as you can. You probably should not use scissors to cut your lawn or use a lawn mower to cut your paper just because you are familiar with one or the other. It is also important to remember that Word and accompanying Office tools were not made for science but for business and a more general usage in mind. It is therefore obvious that Office applications will not suit all scientific needs.

We will start with a basic introduction on how to write a simple document. This covers almost everything you need to know to work with \LaTeX . Once you understand the basics you have the foundation to build more advanced features which is what most of the compendium concerns. By expanding the capabilities you tap into a wealth of features that makes \LaTeX unique. In the end you will expand your own knowledge through experience and most importantly through all the other users who share their experience on e.g. [TeX StackExchange](#)

2 A quick start to writing with \LaTeX

In this section I will provide the basics to produce a document with \LaTeX . When you study this section you are recommended to follow along and try the examples in your own document on Overleaf. Let's dive in!

In Overleaf (Figure 3) you start a new blank project and provide a project name. In my example, I called it 'Example'. Overleaf will then open the document in a four pane display as shown in the figure below. The top left is the *project pane* which displays what files are included in the project. By default the file you receive will be named `main.tex` but this can be changed to something more meaningful. Below the project is the *File outline* pane which will show the document outline. The centre pane is the *\LaTeX code pane* which is where you enter your text and code to create your document. Finally to the right is the *preview pane* which shows how

the document looks.

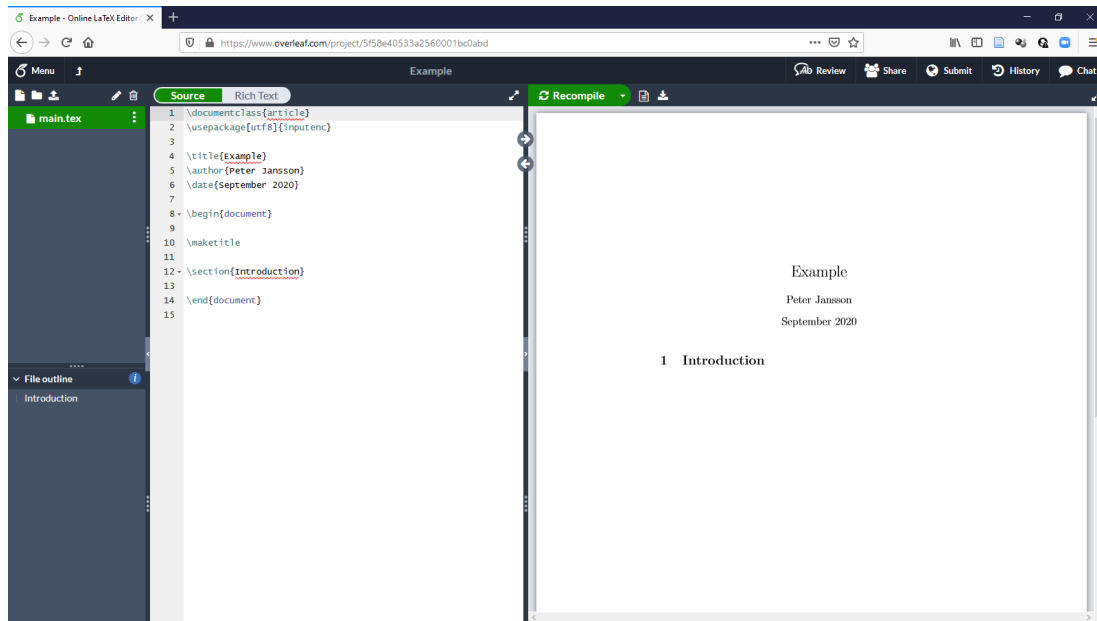


Figure 3. The Overleaf web interface to work with LaTeX projects. For details refer to the text.

At this point it may have become evident that we essentially are programming the document instead of writing in a ‘what-you-see-is-what-you-get’ (WYSIWYG) environment such as that of an ordinary word processor. What you write in the code window must be compiled by \LaTeX to yield the final layout. The right pane is not automatically updated by default. It is possible to change settings so that it does an automatic update. However, this means a lot more compiling and transfer of data will occur and the text you enter will still not appear as soon you type something but with a delay. The compiling process allows \LaTeX to not only consider what comes before a particular point in a document but also what comes later to optimise the layout throughout. This also means that we need to make Overleaf compile our document before we see anything new that we have added to the document. This is accomplished by pressing the green ‘Recompile’ button at the top of the preview pane.

The simple document we receive in Overleaf shows us several things. First you notice is that there are commands identifiable by a backslash and often with an argument within curly braces, such as `\documentclass{article}`. \LaTeX produces documents in response to such commands, similar to how you build a web-page in html. An important point to note is that all commands in \LaTeX are *case sensitive* so make sure you do not accidentally use the wrong case in a command. It is useful to know that all basic \LaTeX commands are written in lower case. You will with time learn many commands but it is still a surprising few that are needed to get you very far.

Let us look at the basic document example:

```
1 \documentclass{article}
2 \usepackage[utf8]{inputenc}
3
4 \title{Example}
5 \author{Peter Jansson}
6 \date{September 2020}
7
8 \begin{document}
9
10 \maketitle
11
```



```

12 \section{Introduction}
13
14 \end{document}

```

We can identify three key commands:

```

\documentclass{article}

\begin{document}

\end{document}

```

These commands are mandatory for all \LaTeX documents and must hence always be present to tell \LaTeX what kind of document to produce (in this case an article; `\documentclass{article}`) and where the document text that should be type set begins and ends (`\begin{document}`–`\end{document}`). The following ‘rules’ apply:

- You are not allowed to place anything before the `\documentclass{article}` command. Doing so will yield an error.
- The space between the `\documentclass` and `\begin{document}` commands is reserved for adding features that add to or modify your document, i.e. other commands. This is referred to as the *preamble* of the document. You are not allowed to place any document text in the preamble.
- The content of your document must be placed between the `\begin`–`\end{document}`. If you place anything after the `\end{document}` it will simply be ignored.

We will get back to this in more detail but the basic rules outlined above must be followed.

In the basic document example we can identify three commands placed in the preamble

```

4 \title{Example}
5 \author{Peter Jansson}
6 \date{September 2020}

```

These commands are allowed to be placed in there. The first command `\title{}` allows you to provide a title for your document and the second `\author{}` allows you to provide your name to the document. The third allows you to enter a date. In my case Overleaf automatically entered ‘September 2020’ because this was when I started the project shown in the figure.

The three commands create the document header we see in the preview pane. However, they will only display if we enter a fourth command `\maketitle` which takes the first three and produces a title for your document. Note that this command must be placed after the `\begin{document}` because it generates the formatted title as part of our document. It is possible to remove the `\date{}` command, `\maketitle` will then extract the current date from the computer and display the date instead of the text provided in the example. What we now have is the header of the document. We now turn to the content. I should point out that the three commands for the title does not have to be placed in the preamble but Overleaf has simply decided to place them there. You can place them with the `\maketitle` command in the main document area.

In the remaining document text you see the command `\section{Introduction}` which produces the heading ‘1 Introduction’. Note that headings are numbered by default. \LaTeX has predefined the look of the headings so that all you have to do to start a new section in your text is to use the `\section{}` command and add your heading text within the curly braces. The heading will always be on a separate line with a pre-defined spacing above and below the heading.

After the heading you can simply type in the text you want to place under that heading. When you do so you will probably notice a few unexpected things.

If you try to start a new paragraph by simply pressing the enter (or return) key ‘Word style’ and continue typing, no new paragraph will start. Instead the text runs on continuously. This is because \LaTeX uses an empty line to identify a paragraph break. So, you need one return to end the line and a second to cause the paragraph break. Adding multiple returns between paragraphs will still result in a normal paragraph break. \LaTeX considers any number of empty lines as a single paragraph break. In other words you cannot create vertical space by entering several returns after each other; another difference from a word processor.

By default \LaTeX uses indentation rather than empty vertical space to signal paragraph breaks. This can of course be changed although it requires a few additional commands. \LaTeX also uses the typographic standard that the first paragraph after a heading is *not* indented but subsequent paragraphs are. The default settings in \LaTeX always follow typographical standards.

One aspect that can cause confusion is that \LaTeX determines all relevant spacing in a document. This means that, for example, if you type in four consecutive spaces \LaTeX still interprets this as one space. This also applies if you try to use the tab key to offset text. It will still result in a single space. So you will not be able to push text around by adding several spaces or tabs as you can in a word processor. We will return to how we can create both horizontal and vertical space in a document if we really need to.

A final command that I have not commented on is `\usepackage[utf8]{inputenc}`. This line asks \LaTeX to load a macro package called ‘inputenc’ (*input encoding*; section 8.12) into your document. The reason for this is that generic \LaTeX does not recognise accented letters. The package allows \LaTeX to understand accented letters such as the Swedish å, ä and ö. If you type in accented letters in a \LaTeX document without having loaded the `inputenc` package, they will simply not be visible. That said, it is possible to get around this problem by choosing how you compile your document, something we will return to later.

Having studied the few basic commands in the Overleaf basic document there is an important point we need to realise. Whenever we use curly braces ({ }) or square brackets ([]) in a command they need to be paired. If you accidentally pair a curly open brace with a close square bracket ({]) or *vice versa* \LaTeX will issue an error and the command will not be executed. So as in any programming language, make sure opening and closing braces, brackets and parentheses in commands match. Unmatched or open pairs are the source of the most common problem when a \LaTeX document does not compile. In Overleaf you always get suggestions for how the commands should be typed and constructed which means you get the correct matching braces and begin-end forms.

Try it

- Add some text of your own and observe what happens if you put many consecutive spaces or tabs in the text and use the return key to make new paragraphs.

Let us now look at some basic features of a document. An occasionally overused but nevertheless useful feature of a document is the *list*. A bullet list in \LaTeX is referred to as `itemize` and is invoked by the following code

<pre>\begin{itemize} \item first item \item second item \item third item \end{itemize}</pre>	<ul style="list-style-type: none">• first item• second item• third item
--	---

As you can see the list created by the `\begin–\end{itemize}` environment. Each item, bullet

point, is created by writing the command `\item` followed by the content of that bullet. There are other forms of lists and you can create your own but we will cover that in section 3.11.

It is possible to nest lists by simply adding and `itemize` environment within another.

Try it

- Make your own list. Try nesting lists.

Once we have made a list we may consider adding a table

```
\begin{tabular}{l c r}
\hline
a & BB & CCC\\
aa & B & CC\\
aaa & BBB & C\\
\hline
\end{tabular}
```

a	BB	CCC
aa	B	CC
aaa	BBB	C

A table is thus created by a `\begin–\end{tabular}` environment. After the `\begin{}` is a set of curly braces showing the letters `l c r`. This tells \LaTeX there will be three columns and shows the alignment for them: left, centre and right in this case.

The table itself is made up of several rows of data. Each row has the form `& & \`. The ampersand indicates a tab separating columns and each row of information is ended by a double backslash, which is the \LaTeX instruction to end a line. So the first row in our table should be understood as ‘a – new column – BB – new column – CCC – new line’.

In the table, you also see the command `\hline` which produces a horizontal line. The basic table we get is not exactly beautiful but we will return to tables in section 4 and how to improve on this. For now you know the principles to make a table.

Try it

- Make your own larger table. Use your own data or try to reproduce something from a journal.

Now let us look at an original strength in \LaTeX , typesetting mathematical notation and equations. Type in the following into your document

```
The definition of  $X_Z$  is
given by
\begin{equation}
X_Z = \sum x_z + \beta^n
\end{equation}
```

The definition of X_Z is given by

$$X_Z = \sum x_z + \beta^n \quad (1)$$

This equation is of course nonsensical but serves the purpose of showing some mathematics. First, you notice that there is code in the sentence: `X_Z` which yields X_Z . The dollar signs signals to \LaTeX that it should switch in and out of *mathematical mode*. \LaTeX works in two modes, mathematical and *text mode*. Certain commands and features only work in either mathematical or text mode, rarely in both. If you use them in the wrong mode they give rise to an error. In this example the underscore `_` is \LaTeX mathematical code for a subscript.

It is very important to remember to keep the dollar signs paired. If you left the code above without a closing dollar sign, `X_Z` , \LaTeX would continue to think everything afterwards until the end of the paragraph is mathematics and eventually yield an error message.

After the sentence there is a construction using the environment `\begin–\end{equation}`. This is the way you create *display mathematics*, equations that are on a separate line and as in this case also centred and numbered. In the equation given by the code `X_Z = \sum x_z + \beta^n` you see a couple of mathematical commands `\sum` which yields the summation sign

and `\beta` which yields the Greek letter β . You also notice the `\textsuperscript` command for superscript, the hat, `\hat`.

More on mathematical typesetting will be discussed in section 7 but for information, use, for example, the [L^AT_EX cheat sheet](#) to see examples of mathematical notation.

Try it

- Create your own equations using the cheat sheet. Try to recreate something you see in a published article or book

A final point is that you can block text from being shown. Just like in programming languages there is a way to create ‘comments’ in a `LATEX` document that is visible in the code window but will not show in the preview or printed document. You can accomplish a comment by entering a percent sign, `%`, in front of the text you want ‘commented out’. So if you write three lines (paragraphs) and comment the second, only the first and third will be visible.

```
This first text is visible
```

```
%This second text is not visible
```

```
This third text is visible
```

This first text is visible

This third text is visible

Note that if you enter a percent sign in front of a paragraph, the entire text until the next carriage return occurs in the text will become invisible. A common error you will make is to add a percentage sign when you write a percentage which will comment out the remainder of the sentence. To get a percent sign you use the command `\%`.

Being able to comment out in the text can be useful if you need to leave a reminder to yourself or if you remove or paste in new text that you do not know if you want to use. In most `LATEX` editors commented text will be shown in a different colour so it will be easily distinguishable from the text that will appear in your final document.

You now have almost all you need to write a complete scientific manuscript, but you obviously wonder about figures (graphics) and references. This is a bit more involved and we will return to these aspects in sections 8.10 and 9.

3 Expanding your manuscript

In the quick start, we briefly touched on several types of formatting in `LATEX`. You can go a long way with what you have already learned to write a scientific document, but what you lack is the knowledge of how to change the layout. It is therefore time to provide a more comprehensive view.

3.1 Types of commands

`LATEX` has three different forms of commands. The simplest form is a command that does not take any argument. An example is `\LaTeX` which yields the formatted word `LATEX`. These commands may produce unwanted effects if you just type them in as they are. If you, for example, type `\LaTeX is` you will get `LATEXis`, where the space between ‘`LATEX`’ and ‘`is`’ disappears. The reason for this is that `\LaTeX` and any simple command needs to signal to `LATEX` that the command is completed. A space is one way to accomplish this but then the space is ‘consumed’ and cannot be used to create a space between words. This means that the space we typed will not be used as an inter-word spacing but just to complete the command and ‘`is`’ will start immediately after the command. So to get the space we need we must remember to provide a completion of

the simple commands. This can be accomplished in several ways. We can add a backslash after the command as in `\LaTeX\ is` or add a pair of curly braces as in `\LaTeX{} is`. If the simple command is followed by a punctuation such as in `\LaTeX. Is` and `\LaTeX, is` the punctuation acts as completion. Forgetting this detail causes many inadvertent typographical errors in \LaTeX .

The second type of command is the command that takes one or more arguments. An simple example of this is `\textit{italics}` which yields *italics*. This type of command uses curly braces, sometimes additional braces for input to the command. An example of a command that can use optional arguments is

```
\documentclass[12pt]{article}
```

In this case the `\documentclass` command takes the optional instruction `12pt` which will produce a document with 12 pt text as default size. By default your document will be set using 10 pt text. Note that optional arguments occur within square brackets. Which commands that have optional arguments and knowing what arguments are possible is something you need to learn by looking up the command online.

These commands do not need any completion as discussed for the first type since the braces provide the information \LaTeX needs to understand that the command is complete. A classical mistake with these types of commands is to forget to add the closing brace which then will yield an error.

The third type of command is the ‘environment’ `\begin–\end{}` where the command has an opening and a closing statement and where the argument is everything you place between the two. We have already looked a three such environments `document`, `itemize` and `equation`. In this case it is important to match any `\begin` with a matching `\end` statement.

It is important to know how the different commands should be written and what may be optional arguments. But you will build your knowledge as you continue working with \LaTeX .

3.2 The document structure

In section 2 we saw that the basic \LaTeX document consisted of the following three lines of commands

```
\documentclass{article}
%preamble
\begin{document}
%document text
\end{document}
```

and we also began writing our text in the `\begin–\end{document}` environment. Although we briefly touched upon this structure in the quick start it is worth reiterating the significance again.

First, the `\documentclass` command must be the first command encountered in a \LaTeX document. It may be preceded by comments but not by any other command or plain text.

Second, all text that we want displayed in a document must occur between the `\begin–\end{document}`. Anything you write after the `\end{document}` will be ignored. This can be quite useful since you can move the `\end{document}` up into the text in order to compile only a portion of the text file. It also means you can cut and paste bits of text into your document file after the `\end{document}` for later use.

Third, the space between `\documentclass` and `\begin{document}` is usually called the *preamble* and is reserved for \LaTeX code that is used to influence the behaviour of what you write in the document environment. You will see how this part of the document usually gets filled with several useful tools. It is however, important to realise that no ordinary text can be written in this part of the document, only commands. Errors will otherwise occur.

3.3 Document classes

When you begin writing a \LaTeX document, you need to select a class or type of document. Most commonly we find ourselves working with ‘article’ as we have already seen, but, there are other classes that are standard in \LaTeX

Class	Description
article	Common scientific article
book	Book with chapters and parts
letter	Letters with letterheads
report	Report with chapters
slides	Overhead slides

Apart from the standard classes, there are numerous other classes available. Many are written by enthusiasts for particular purposes such as [CV](#), [sheet music](#), [sudoku](#), etc. Other classes have been developed by publishers and journals to facilitate authoring in journal specific formats. For scientific publishing, the latter is probably what we should look for.

To provide some examples, the [American Geophysical Union](#) and the Open Access publisher [Copernicus](#) has classes for their journals. Most journals on, for example, Wiley, Springer and Taylor & Francis also have class files for their own style. Note that classes may contain new or redefined commands which can lead to unexpected behaviour if you do not carefully check how the class works. In most cases, you will not notice any differences but not all journal classes may be up-to-date or well written.

Try it

- Look at some key journals in your field and locate their class files. Or, search Overleaf for journal templates. You may want to try them out when you feel ready. A word of warning, not all journal classes are well written and straight forward to use.

3.4 Reserved characters

When you write text in \LaTeX there are a number of characters that are *reserved* because they are used as part of commands and instructions for \LaTeX . These cannot be used directly in the text without causing either an error or some unexpected result. You have already come across most: the comment percent sign, the tab stop ampersand in tables and the mathematical mode switch dollar sign as well as super and subscripts hat and underscore signs. In the table below you also see the commands you need to use if you wish to use these reserved characters in the text.

Character	\LaTeX use	Command
%	comment	$\backslash\%$
&	tab stop	$\backslash\&$
\$	mathematical mode	$\backslash\$$
^	mathematical mode superscript	$\backslash\wedge\{ \}$
_	mathematical mode subscript	$\backslash_$
{ }	start stop grouping	$\backslash\{ \}$
~	non-breaking space	$\backslash\sim\{ \}$
#	Parameter in macros	$\backslash\#$

3.5 Type face styles

As with all word processors, basic text is typed out with the normal type face. If you need to change this there are several different choices summarised in the following table

Type	Command
<i>italics</i>	<code>\textit{}</code>
bold face	<code>\textbf{}</code>
SMALL CAPS	<code>\textsc{}</code>
<i>slanted</i>	<code>\textsl{}</code>
sans serif	<code>\textsf{}</code>
typewriter	<code>\texttt{}</code>
<u>underline</u>	<code>\underline{}</code>

The size of the text can also be varied but not in the way you may think. L^AT_EX scales fonts relative to the ‘normal size’, that is the size of the main text. Thus in order to get larger or smaller font sizes you can use the following set of commands, relative to the ‘normal size’.

Font size	Type	Example
5	6	<code>\tiny</code> the quick brown fox
7	8	<code>\scriptsize</code> the quick brown fox
8	10	<code>\footnotesize</code> the quick brown fox
9	11	<code>\small</code> the quick brown fox
10	12	<code>\normalsize</code> the quick brown fox
12	14	<code>\large</code> the quick brown fox
14	17	<code>\Large</code> the quick brown fox
17	20	<code>\LARGE</code> the quick brown fox
20	25	<code>\huge</code> the quick brown fox
25	25	<code>\Huge</code> the quick brown fox

The benefit of this is that as you change the global font size all relative scaling will also change.

The default type size for L^AT_EX is 10pt. The basic sizes you can use are 10, 11, and 12 pt and this can be set using the `\documentclass` command as an *optional argument*. If you want to make an article in 11pt you need to provide

```
\documentclass[11pt]{article}
```

at the beginning of your document. Here the square bracket indicates that the command takes an optional argument stated within the brackets. If you do not specify any optional argument L^AT_EX will use a default value.

Changing type faces is not necessarily straight forward; I am almost tempted to state you should not even try, yet. The reason for the hesitation is that while L^AT_EX typesetting has remained more or less constant since its infancy, fonts have not. This is because as computers have gone from 8-bit to 64-bit and capacity has sky-rocketed, Fonts have developed from containing a basic 128 character ASCII set to now over a million characters. With the backwards compatibility of L^AT_EX it has not been possible to develop with the new standards but rather find ways to implement the new while keeping the old.

In Overleaf, the default is an implementation of L^AT_EX called pdfL^AT_EX. This directly generates pdf files from your L^AT_EX source and is probably the most widely used form of L^AT_EX. In the core L^AT_EX distributions (incl. Overleaf) many fonts are installed that work with pdfL^AT_EX. This is also where the `inputenc` package comes in. The package allows you to work with more

than the basic ASCII characters in a document. The package is in other words a requirement when using pdf \LaTeX .

Another implementation called Xe \LaTeX was developed to make use of the newer Unicode type faces containing over a million characters. If you run Xe \LaTeX you can quite easily access your system type faces (the same you access in Word). This is quite tempting but remember that in order for someone else to run your document, they need to have the same type face. So there are benefits and pitfalls with going either way. We will leave the type face question for now.

In section 2 we had issues with accented letters. This problem is easily circumvented by switching to the Xe \LaTeX compiler in Overleaf since we can then access such accented letters the way they are entered from our computer keyboard. So if you are writing in languages that use accented letters it may be useful to use Xe \LaTeX instead of pdf \LaTeX .

That said, I can mention the [\$\LaTeX\$ Font Catalogue](#), which contains an overview of generally available free type faces and also how to make them usable in your document. An important point to make is that only certain type faces have support for mathematics. These are identified in the font catalogue. If you use a type face that does not include mathematics anything written with mathematics will be written using the default font. In some cases you will probably barely notice the difference but in other cases it will just be ugly. The freedom to chose can thus be quite limited. At the same time there are good reasons not to experiment too much with type faces since we are not professional type-setters.

You can try to change fonts by following the instructions in the font catalogue and try other solutions found on \TeX StackExchange.

Try it

- Try the different ways to change font and size, including the switch between 10, 11, and 12 pt in the `documentclass` command.

3.6 Typographic features

\LaTeX contains many special characters and features. A list of some of these is

Feature	Command	Function
‘ ’	single grave accent and apostrophe	British open–close quote
“ ”	double grave accent and apostrophe	American open–close quote
–	--	en-dash
...	<code>\ldots</code>	–
©	<code>\copyright</code>	–
®	<code>\textregistered</code>	–
\TeX	<code>\TeX</code>	–
\LaTeX	<code>\LaTeX</code>	–

Most of these need no further comment but I will point out a couple of details. First, when it comes to using quotes, it is possible to use the common double quote on the keyboard but it does not look very good. In addition, double quotes in English is American typography so unless you are American or come from a country that uses the American typesetting standard, you should stick to the British English single quote.

Second, a common problem with texts is that authors do not know the difference between a hyphen (dash) and a longer dash (en-dash, --). The hyphen is part of your key board and used when you hyphenate words or put two words together. The longer dash is used when you want the meaning to be ‘to’ as in 4–7 (four to seven). The longer dash looks like a normal minus sign but \LaTeX actually have a special minus sign produced in math mode, compare – (en-dash) and

– (math minus, \$- \$). Since it is very easy to distinguish between the different types of dashes in \LaTeX you should make an effort to use them appropriately.

3.7 Breaks and space

When you write text you need to occasionally deliberately break the text or words to fit the space available. One of the strong points of \LaTeX is to automatically manage these duties but it will never be possible to automate typesetting to 100%.

Let us start by looking at a few commands that are useful for breaking text over pages and lines.

Command	Function
<code>\newpage</code>	page break
<code>\pagebreak</code>	page break keeping justification
<code>\\</code>	line break without new paragraph
<code>\newline</code>	line break without new paragraph
<code>\linebreak</code>	line break keeping justification
<code>\nolinebreak</code>	–
<code>\noindent</code>	prevent indentation of a paragraph
<code>\indent</code>	indent a non-indented paragraph
<code>\-</code>	conditional hyphenation
<code>\</code>	keep normal space
<code>\@</code>	end of sentence space after capital letter

Most of these are quite self-explanatory and you can test their function in your own text.

The conditional hyphen and hyphenation as a whole may deserve a comment. If you end up with a word that is too long and it does not seem to hyphenate at all or perhaps wrong (\LaTeX uses a set of rules for hyphenation), then you can insert a conditional hyphen. If we take the word ‘multispectral’, we can prepare the word with conditional hyphens so that \LaTeX will know where breaks can occur, the code will be `mul\ -ti\ -spec\ -tral`. These conditional breaks will not show unless used.

If you use a word, for example, a complex scientific term that is unlikely to be hyphenated correctly you can provide \LaTeX with information how to treat the word in the text. By placing

```
\hyphenation{mul\ -ti\ -spec\ -tral}
```

in the preamble, you pass this information to the hyphenation engine to correctly deal with the word any time it is encountered in the document. It is also possible to tell \LaTeX to not hyphenate a word by adding the word in the same command but without any hyphenation marks. So

```
\hyphenation{Fortran}
```

will result in the word ‘Fortran’ never to split across two lines.

By default \LaTeX uses English hyphenation rules. If you write in another language you may therefore get word breaks that do not fit that language. You can correct this problem by using a language package called ‘Babel’ which we cover in section 8.3.

\LaTeX uses an intricate system to keep track of good word spacing. By default you will get a slightly wider space after a period, indicating the end of sentence and beginning of a new. This rule is, however, not always correct. If we have a lower case abbreviation with periods each period will be interpreted as an *end-of-sentence*. By adding a backslash after the period in the abbreviation we get normal word spacing after the period. If we on the other hand have an abbreviation consisting of capital letters followed by a period, the period is not considered an

end-of-sentence but part of the abbreviation. So if the sentence ends with such an abbreviation, then we must add a `\@` to make \LaTeX aware that the period after the capital letter abbreviation is the end of the sentence and produce a wider space. The following table shows how we can correct problematic spacing originating from abbreviations.

Example	Code
J. Geophys. Res.	J. Geophys. Res.
J. Geophys. Res.	J. Geophys.\ Res.
in UNESCO. Another	in UNESCO\@. Another
in UNESCO. Another	in UNESCO. Another

Lastly, in the old days of monospaced typewriter fonts, it was customary to insert two spaces between sentences. You sometimes see people still do this in word processors. \LaTeX inserts a space that is slightly longer than a single space, but still shorter than two, by default. It is possible to get single space between sentences by adding the command `\frenchspacing` at the beginning of the document. You can turn the French spacing off by adding `\nonfrenchspacing`. Since adding two or more spaces between sentences in \LaTeX does not affect the spacing, it does not matter if you accidentally hit the space bar a couple of times extra, the only way to really easily affect the spacing is through the `frenchspacing` commands.

Try it

- Test the different commands affecting spacing within your own text so that you understand the difference they make. It may be beneficial to work on several identical paragraphs in parallel so that you can visually compare the results.

3.8 Headings

The headings in \LaTeX come in two flavours, numbered (default) and unnumbered. The unnumbered versions can be obtained by using so called *starred* versions of the command. The section command `\section{}` is thus converted to unnumbered by writing `\section*{}`. Some command can be modified using their so-called starred versions, but note that this is not a general feature but only applies to specific cases.

Numbered	Unnumbered
<code>\part{}</code>	—
<code>\chapter{}</code>	<code>\chapter*{}</code>
<code>\section{}</code>	<code>\section*{}</code>
<code>\subsection{}</code>	<code>\subsection*{}</code>
<code>\subsubsection{}</code>	<code>\subsubsection*{}</code>
<code>\paragraph{}</code>	<code>\paragraph*{}</code>
<code>\subparagraph{}</code>	<code>\subparagraph*{}</code>

The levels `part` and `chapter` are only available in the book class and will not work in the article class.

It is possible to produce special numbering for appendices by using the command `\appendix` where the appendices start. This will force \LaTeX to start alphabetic sectioning numbering instead of numeric, which is the common form for an appendix. In addition, there is also a package called `appendix` which provides additional control of appendices.

There are two commands called `\frontmatter` and `\mainmatter` that numbers pages with roman numerals from the front-matter command until the main-matter command. After that

numbering will be with Arabic numerals. This is common in type setting books where pages with roman numerals are used for pages containing material that is not part of the main book content such as table of contents, preface etc.

Try it

- Try the different forms of headings. Note, however, that for some to work you need to switch your document to the book class.

3.9 Table of contents

To create a table of contents is very simple in \LaTeX . You need to do one thing and that is to type the command `\tableofcontents` where you wish the table of contents to appear. There is, however, one caveat. The table of contents will only be automatically generated from the numbered sections. If you use starred heading commands you will not see a table of contents. This can be used for special purposes but also remedied to obtain a table of contents for starred heading commands.

If you, for example, have numbered headings in your document and edit in a star in one of the heading commands, that heading will no longer be visible in the table of contents. There is a work-around for this by changing the way the table of contents work. If you add

```
\setcounter{secnumdepth}{0}
```

to the preamble this tells \LaTeX not to number any sections. By adding or removing this line from your file you can essentially turn numbering on or off.

Another way to accomplish this is to add

```
\addcontentsline{toc}{type}{heading title}
```

after each heading. `type` refers to the heading level. If you for example have an unnumbered section (e.g. Acknowledgement or References) that should be in the table of contents you will provide the following after each heading

```
1 \section{The section heading}
2 \addcontentsline{toc}{section}{The section heading}
3
4 \subsection{The subsection heading}
5 \addcontentsline{toc}{subsection}{The subsection heading}
```

The `\addcontentsline` command tells \LaTeX to add information about a section to a file that it keeps to create the table of contents. The first argument is the name of the file. This will be a file named the same as your document but with the extension `.toc`. The second argument tells \LaTeX what level of heading you add to the file. The third is the heading text as it will appear in the table of contents.

In \LaTeX you can also provide similar table of contents for your figures and tables using the commands `\listoffigures` and `\listoftables`. We will get back to details how this can be accomplished in section 6.

If you have many sections of all levels it may make sense to remove the lowest level (subsubsection) from the table of contents in order to shorten it. By using the `\setcounter` command to change the table of contents ‘depth’ (`tocdepth`), i.e. how many sub levels should be visible in the table of contents, as follows

```
\setcounter{tocdepth}{1}
```

is then useful because it allows you to limit the ‘depth’ of the table of contents. With the argument ‘1’ only sections and subsections will be visible.

3.10 Index

The capability of making an index in your document is probably not high on your list of features for everyday use. But, if you intend to work on a larger text such as a book, indexing is a very useful feature. In order to create an index in \LaTeX you need to use a packages to extend the functionality, `imakeidx` (Gregorio, 2016).

In order to create an index in your document you need to add the following to your preamble

```
\usepackage{imakeidx}  
\makeindex
```

and then add

```
\phantomsection  
\printindex
```

at the location in your document where you want your index to appear. In order to have the index appear in the table of contents you need to add the command `\phantomsection`. The reason for this is that \LaTeX does not identify the index provided by the package as a section since it is not a generic part of \LaTeX unlike, for example, the section, subsection, chapter, etc. commands which all are automatically recognised.

In order to add items to your index you use the `\index{}` command. You simply enter this command after the word you wish to add to your index and enter the index item name as an argument. So to add the word ‘index command’ to the index you need to type e.g.

```
In order to add items to your index you use the index  
command.\index{index command}
```

This creates an index entry ‘index command’ followed by the page number where it is located. You can use the index command for the same entry in several places in a document. You will then get a list of page numbers where the index entry appears.

It is possible to obtain more complex entries using additional notation. When you have an index entry that occurs in multiple places it is common that there is one location that can be considered the main occurrence. This can be the location where it is explained in more detail while other occurrences are subordinate. In such a condition you can make that key entry bold by using the following notation

```
In order to add items to your index you use the index  
command.\index{index command|textbf}
```

The entry from this index command will then appear in bold in the index. As you can see text formatting is handled by adding a `!` between the index entry and the formatting instruction. There are other formatting features that you can use but please refer to the `imakeidx` package documentation.

Another feature that is often useful is to be able to add themes to an index. In this text I, for example, often mention packages. So a typical entry in the index would be ‘`imakeidx` package’ which is listed under ‘I’. But if you want to look up a package and you do not know

its name it would be useful to have all packages listed under a main entry ‘packages’. This can be accomplished by using the following formatting of the index command

```
\index{package!imakeidx}
```

If you enter similarly formatted index entries where you mention other packages you will find them alphabetically ordered as a secondary term under the main index entry ‘package’. This way you can provide a very useful level of information to the index. Note however that you may still want an entry ‘imakeidx package’ which involves making several index entries at the same location such as

```
\index{imakeidx package}\index{package|imakeidx}
```

3.11 Environments

In the quick start we came across several environments. When you enter text there are several more that you should be aware of. Common to all is that they use the environment structure `\begin{}`–`\end{}`.

Environment	Action
<code>center</code>	centers content
<code>flushleft</code>	flushes content left
<code>flushright</code>	flushes content right
<code>quotation</code>	decreases text area width for a quote with indentation
<code>quote</code>	decreases text area width for a quote without indentation
<code>verse</code>	for poetry
<code>minipage</code>	produces a separate environment with its own dimensions
<code>itemize</code>	produces bullet lists
<code>enumerate</code>	produces numbered lists
<code>description</code>	produces lists with key words as ‘bullets’
<code>verbatim</code>	reproduces all text including protected symbols as written

Most of these environments are self-explanatory and something you can explore on your own.

The `minipage` environment is worth looking into a bit deeper. In this text I have used the `minipage` to create ‘side-by-side’ looks at code and result. The following display

<code>\begin{itemize}</code>	• first item
<code>\item first item</code>	
<code>\item second item</code>	• second item
<code>\item third item</code>	
<code>\end{itemize}</code>	• third item

is created by

```

1  \noindent\begin{minipage}[c]{0.4\textwidth}
2    \begin{verbatim}
3      \begin{itemize}
4        \item first item
5        \item second item
6        \item third item
7      \end{itemize}
8    \end{verbatim}
9  \end{minipage}\hfil
10 \begin{minipage}[c]{0.4\textwidth}
11   \begin{itemize}

```

```

12     \item first item
13     \item second item
14     \item third item
15 \end{itemize}
16 \end{minipage}

```

This may seem daunting but if you look closely, it consists of two `minipage` environments typeset side by side (no paragraph break) and where the mini-pages are set to be 40% of the text width (`0.4\textwidth`) each. What is not obvious from the typesetting is that I have pushed the two mini-pages apart by inserting the command `\hfil` which is a basic command for inserting space if there is extra space to be had. There are a whole series of such commands but they are beyond the scope of this text. I can only urge you to start a little self study on the more advanced topics that allows you to dive into the interior of \LaTeX .

The `description` environment is very useful and works just like the `itemize` and `enumerate` environment but with the exception that you need to provide an argument.

<code>\begin{description}</code>	1 first item
<code>\item[1] first item</code>	
<code>\item[two] second item</code>	two second item
<code>\item[III] third item</code>	
<code>\end{description}</code>	III third item

It is also possible to change the look of all list environments. A simple way to change the bullets in `itemize` is to add an argument after `\item[]`. If you want an en-dash instead of the bullet you simply write `\item[--]` for each of the items you want to list.

Try it

- Try the different environments. In the case of lists, try to nest different types to see the effects of nesting on their behaviour.

3.12 cross-referencing

\LaTeX comes with a powerful cross referencing system. The system works on the principle that you put a label on everything you want cross-referenced and then use a command to take information about that label and insert it into the text. The following are the commands you may encounter.

Command	Action
<code>\label{}</code>	establishes a label
<code>\ref{}</code>	takes a label and replaces it with a number
<code>\eqref{}</code>	specific references to equation numbers
<code>\pageref{}</code>	specific references to page numbers
<code>\cite{}</code>	generic literature references (see section 8.16)

The `\label` command is used to assign a unique name (label) to a section, figure, table or equation. You need to provide a name for the label and this can be any name you chose. If we, for example, have a map figure in our manuscript, I would label this figure `\label{fig:map}`. First, I would recommend that you have a descriptive word in the label so that you know what that figure is. Second, I prefer to add a ‘tag’ `fig:` for figure, `tab:` for table, `eq:` for equation and `sec:` for section so that I, for example, do not confuse a tag for a figure with one for an equation. You can come up with your own scheme, however, you will be limited to letters a–z,

numbers and basic punctuation for your labels. So no protected symbols or accented letters can be used. In Overleaf the tags mentioned above will be automatically suggested within the figure, table and equation environments.

Once you have assigned a label to an object you can reference, for example, the map figure we labelled above as `\ref{fig:map}`. In the resulting layout \LaTeX will take the number of the figure labelled `fig:map` and insert that instead of the `\ref{fig:map}` command. The benefit of this is that all figures (and tables and equations) are numbered implicitly, which means that if you rearrange figures the labels will be assigned new appropriate numbers and so will all your references in the text. You simply do not have to worry.

There are a set of specific reference commands. `\eqref{}` takes only labels from equations but you can just as easily use the `\ref{}` command. The `\pageref{}` will display the page number where your label is defined. This means that if you have labelled your map somewhere and wish to refer to it, not by its figure number but the page on which it occurs, then you should use `\pageref{fig:map}`.

Finally, \LaTeX comes with a cross-referencing system for citations and reference lists. I will not go into any details here because there is more to the system than just cross-referencing. Please refer to section 8.16 for a detailed description. At this point it is sufficient to know that the `\cite` command exists for literature references coupled to a basic reference list.

3.13 Special commands

\LaTeX has several built in commands that perform tasks you want to use. Here is a list of ones we will look more carefully at

Numbered	Unnumbered
<code>\today</code>	provides current date
<code>\the</code>	allows typesetting of internal variable values from \LaTeX
<code>\year</code>	variable containing current year
<code>\day</code>	variable containing current day
<code>\month</code>	variable containing current month
<code>\footnote{}</code>	typesets footnote
<code>\marginpar{}</code>	adds text in the margin where the command occurs

\LaTeX can take information from the computer and display in documents. The most common command for this is `\today` which displays the current date. When you created the title of the document with `\maketitle` \LaTeX implicitly uses the `\today` command to add a date to the title. You can use the command anywhere if you want to have the current date in your document but do not confuse this with a fixed date, it will be updated every time you run your document.

A usually little known but powerful command is `\the`. This command takes a value from some parameter within \LaTeX and displays it as text in your document. Since \LaTeX has variables for year, month and day called (of course), `\year`, `\month` and `\day`, it is possible to extract that information into the text by using the combination of commands `\the\year` etc. But, this will only yield the numbers, not, for example, the month in text. It is however, possible to program \LaTeX to convert the number to text.

Footnotes are rarely used in scientific articles but is of course a possible ingredient in a text. \LaTeX has a footnote command that will take the text you enter as an argument and place it at the bottom of the page like in the example here³. To create a footnote you simply add the command `\footnote{}` right where you want the reference to occur. \LaTeX will insert an index number and add the footnote at the bottom of the page automatically including a dividing line separating the footnote from the text.

³This is footnote text entered in the paragraph on footnotes above

The `\marginpar{}` produces the text given as the argument as a paragraph in the margin of the document. This is useful if you want to, for example, add visual reminders to your self in the text. I find it useful to define a new command (see section 11) based on the `\marginpar` where the type size is smaller and perhaps also in some colour to make it more visible. It can be used both for reminders but also as a tool in the text. We can for example type

```
\marginpar{\scriptsize\texttt{\textbackslash marginpar}}
```

in our text next to the first mention of the command in the paragraph to generate the script-sized command in the margin so that we can easily find where it is mentioned in the text (as is done in this section). The margin paragraph will start next to where it is located in the text.

4 Tabular information

Typesetting good tables is not easy anywhere. Quite often we see tables made to look like a heap of boxes. This is not good type-setting practise. We will thus focus on simple publication quality tables.

In \LaTeX , the basic environment for creating tables is called `tabular` and a simple table would be made as follows (with output to the right):

```
\begin{tabular}{l c c}
\hline
1 & 2 & 3 \\
\hline
A & B & C \\
a & b & c \\
\hline
\end{tabular}
```

1	2	3
A	B	C
a	b	c

This is neither sophisticated nor beautiful but shows the principles. Obviously the `tabular` uses a `\begin`–`\end` structure. After the `\begin{tabular}` is a set of curly braces containing the letters `l c c`. These letters tells \LaTeX how to align the content in the three columns, first column `l` for left-adjusted and the other two `c` for centred. There is of course an `r` for right-adjusted columns.

The core of the table contains rows such as for example `1 & 2 & 3 \\\`. We can see that the ampersand is used to mark columns 2 and 3 but is not used for column 1. This is how \LaTeX uses the ampersand in the `tabular` environment, it indicates a jump, tabular stop, to the next column. The line ends with a double backslash which tells \LaTeX that the line is complete and to switch to a new line. All lines of data in a table must end with the double backslash. I have also inserted some horizontal lines using the `\hline` command. This code does not take a double backslash to mark that it is on a separate line.

The example shows us a few things worth noting. First, that a table should be as simple as possible when it comes to lines, this is good typography. Second, \LaTeX will determine the width of columns and the width of the table based on the content. This is of course fine as long as our data and column headers are simple and short.

By loading an package called `booktabs` (see section 8.4) you get access to a series of additional lines, called rules that will help you organise more complex tables. In the following example we use the non-native commands `\toprule`, `\midrule`, `\bottomrule` and `\cmidrule`. They are probably all self-explanatory except the last.

`\cmidrule[](){}` takes three arguments (note the different types of brackets used). The first specifies the thickness of the line and the second if it should be trimmed. Both the thickness and trim arguments are optional. If not trimmed, the lines will go the full width of the columns and possibly meet with neighbouring lines. Trimming can be achieved by entering `1` for left trim

and r for right trim or both into the argument. The third argument takes the column numbers across which the rule should span.

As an example we can look at this small but relatively complex table.

Year	b_w (m w.e.)	H_{ss} (m)		$b_w(k)$ (m w.e.)		S_{wi} (%)	$b_c(k)$ (m w.e.)		$b_a(k)$ (m w.e.)	
		29	28S3	29	28S3		29	28S3	29	28S3
1997/98	1.35	2.7	2.4	0.03	0.06	6	0.06	0.06	0.09	0.12
						7	0.07	0.07	0.10	0.13
						8	0.08	0.09	0.11	0.14
1998/99	1.33	2.9	2.6	0.03	0.06	6	0.06	0.08	0.09	0.13
						7	0.07	0.09	0.10	0.15
						8	0.08	0.10	0.11	0.16

was created by

```
{\begin{tabular}{l c c c c c c c c c c}
\toprule
Year & & & & & & & & & & 
&\multicolumn{2}{c}{ $H_{ss}$ }& 
&\multicolumn{2}{c}{ $b_w(k)$ }& 
&  $S_{wi}$  & 
&\multicolumn{2}{c}{ $b_c(k)$ }& 
&\multicolumn{2}{c}{ $b_a(k)$ }\\
\cmidrule{rl}{3-4}
\cmidrule{rl}{5-6}
\cmidrule{rl}{8-9}
\cmidrule{rl}{10-11}
&&29&&28S3&&29&&28S3&&29&&28S3\\
&&(m w.e.)&&\multicolumn{2}{c}{(m)}&&\multicolumn{2}{c}{(m w.e.)}&&(\%)&&\multicolumn{4}{c}{(m w.e.)}\\
\midrule
&&&&&&6&0.06&0.06&0.09&0.12\\
1997/98&&1.35&&2.7&&2.4&&0.03&0.06&7&0.07&0.07&0.10&0.13\\
&&&&&&&&&&8&0.08&0.09&0.11&0.14\\
&&&&&&&&&&6&0.06&0.08&0.09&0.13\\
1998/99&&1.33&&2.9&&2.6&&0.03&0.06&7&0.07&0.09&0.10&0.15\\
&&&&&&&&&&8&0.08&0.10&0.11&0.16\\
\bottomrule
\end{tabular}
```

An important command we encounter here is the `\multicolumn{}{}{}` command. The command places a single entry across any number of columns. It takes three arguments. The first is the number of columns it should span. The second is the alignment we want. The third is the text that should be placed across the multiple columns.

When you work with tables it may be advantageous to use Excel to organise the columns and rows of information. The add-in `Excel2LaTeX.xla` can be found on [CTAN](#). The file is opened in Excel and provide you with new features in the Add-in tab. With this add-in you can generate a table in Excel and convert it to \LaTeX tabular code to be included in your document.

Try it

- Typeset some of your own tables. Start with a simpler table.

5 Graphics

To include graphics files into a \LaTeX document we need to add a set of macros that can accomplish this; adding graphics is thus not part of generic \LaTeX . To enable graphics we need to add a package called `graphicx` (section 8.10) by adding:

```
\usepackage{graphicx package}
```

to our document preamble. This then allows us to use the command

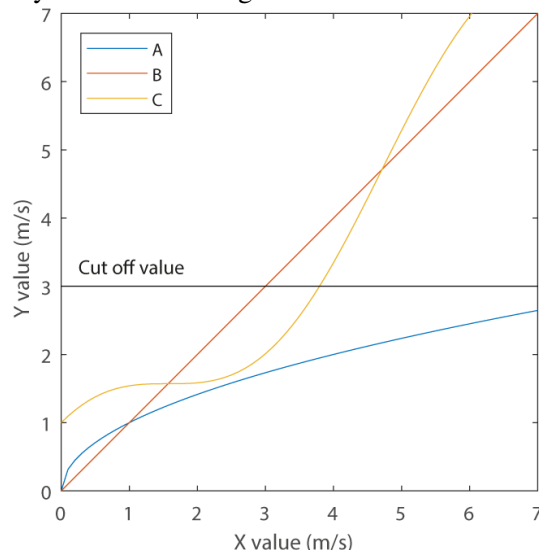
```
\includegraphics{filename}
```

to include a graphics file.

In Overleaf I prefer to keep included graphics separate from other files belonging to the project so I usually add folder called `fig` to the project and add my graphics files there. So when I want to use a graphics file I will write, for example:

```
\includegraphics{fig/BasicPlot.png}
```

to yield the following:



It is also possible to add the command `\graphicspath{ }` to the preamble and specify the paths where \LaTeX should look for graphics. So in my case I can add

```
\graphicspath{{fig/}}
```

to the preamble and then you do not need to add the path to the file in the graphics file call using the `\includegraphics` command. It is also possible to avoid providing the graphics file extensions in the `\includegraphics` command by placing the `\DeclareGraphicsExtensions` command in the preamble to define graphic file extensions, such as:

```
\DeclareGraphicsExtensions{.pdf,.png,.jpg}
```

The `\includegraphics` command can handle JPEG, PNG, and PDF when we compile with pdf\LaTeX . Note that TIFF bitmap files cannot be included directly but it is possible if we include the TIFF in a pdf file. This can be accomplished by either opening the file in a graphics software that can save the graphics in pdf format or by using a pdf software to ‘convert’ the TIFF image to a pdf. This process may require some trial and error to ensure that the TIFF file is reproduced correctly in the conversion. To learn more about \LaTeX graphics I recommend visiting this

[TeX.stackexchange](#) post on what graphics formats can be included in L^AT_EX and what limitations exist. For SVG graphics there is also the [SVG package](#). Because the L^AT_EX system is public new solutions may come about so it is always good to occasionally do a search to see what is new.

6 Floats; tables and figures on the run

L^AT_EX provides two special environments for figures and tables that have the property that they can be moved by L^AT_EX from the location where you have entered them to a typographically preferable place. This feature is called a float because they may ‘float around’ in the document. Many beginners can find this feature frustrating, not knowing exactly before hand where the figure may appear. But, in this case, L^AT_EX has much better knowledge of good layout and typesetting than the average user so the irritation is just a lack of knowledge.

To make a float you use the two environments `figure` and `table`. For a figure you would use the following code

```
1 \begin{figure}[ht]
2 \includegraphics{filename}
3 \caption{This is the figure caption text}
4 \label{fig:x}
5 \end{figure}
```

The figure environment should contain three commands. First there is the `\includegraphics{}` command which brings in a graphics file into the document. Second is a new command called `\caption`, specific to floats, which produces a caption using the text you enter as the argument to the command. This caption should always be typeset below the figure graphics, which is why we place the commands in that order, and the figure–caption pair will never separate across pages etc. Third is the `\label` command which allows you to refer to the figure using the label name (see section 3.12).

A table follows the same basic recipe but the table itself is made up of a tabular environment (section 4)

```
1 \begin{table}[ht]
2 \caption{This is the table caption text}
3 \label{tab:x}
4 \begin{tabular}{c c}
5 . . .
6 \end{tabular}
7 \end{table}
```

In this case we place the caption before the tabular environment because the table caption must go above the table. We also have a label working the same way as for the figure environment.

Creating these floats is thus quite simple. It is, however, possible to influence the placement of the floats in the document. To do this, we can add an optional argument to the `\begin{}` command. This optional argument can be one or a combination of

Option	Action
<code>h</code>	<i>here</i> , approximately where it occurs in the source text
<code>t</code>	<i>top</i> of the page
<code>b</code>	<i>bottom</i> of the page
<code>p</code>	on a special floats <i>page</i>
<code>!</code>	Override internal parameters for ‘good’ float position
<code>H</code>	<i>Here</i> , precisely at the location in the code, similar to <code>!ht</code> ¹

¹This requires a package called ‘float’, section 8.7

The way float placement works has been described by [Frank Mittelbach on T_EX StackExchange](#). and I will summarise the flow here.

When a float is encountered, L^AT_EX attempts to place it immediately according to its rules. If this does not succeed, then L^AT_EX places the float into a holding queue to be considered when the next page is started. Once a page is completed, L^AT_EX tries to place remaining floats in the holding queue as best as possible. To do this it will first try to generate as many float pages as possible. If this is not possible, it will try to place the floats into top and bottom of pages. It looks at the remaining floats and either places them or defers them to a later page. After that, it starts processing the text for the page. In the process, it may encounter further floats and the process goes on. If the end of the document is reached before the queue is emptied, L^AT_EX will add pages and dump remaining floats there.

As you can tell the process is quite involved and never random. If there is need to try to create a page of floats inside a document, you can use the command `\clearpage` which will empty the queue onto pages before continuing with the document.

The option H is not generic L^AT_EX and requires additional work which we will discuss in section 8.7.

Try it

- Test the different float environments with a simple figure and table and change the positioning arguments to see their effects. Note that you need to have a few pages of text with plenty of paragraph breaks to be able to really test the floats. I recommend loading the `lipsum` package (section 8.14) and adding text by typing `\lipsum[n]` where *n* can be a number or a range (e.g. 2-5). Check the package documentation for details; `lipsum` is useful if you need text to see how a layout works.

7 Mathematical typesetting

Since mathematics is one of L^AT_EX original strong points there are too many introductions to count on writing mathematics out on the Internet. I will therefore not try to add another one to the mix but draw up some basics that will allow you to go along way and form a basis for your own research. As we saw in the quick start, mathematics come in two flavours, in line and display mathematics. The same commands are used in both cases in order to produce the mathematical notation so we will first focus on the different forms for displaying mathematics and then onto the details of how to create the equations.

The basic display mathematics environment is `equation`. As we have seen earlier it works just like any other environment

```
\begin{equation}\label{eq:x}
a^2 = b^2 + c^2
\end{equation}
```

$$a^2 = b^2 + c^2 \quad (2)$$

In this example I have also added the label which means you can now cross-reference the equation number as `\ref{eq:x}` in the text. There is also an environment for sets of equations

```
\begin{eqnarray}
c_1 &=& a_1x + b_1x \quad \label{eq:a} \\
c_1 &=& a_2x + b_2x \quad \label{eq:b}
\end{eqnarray}
```

$$c_1 = a_1x + b_1x \quad (3)$$

$$c_1 = a_2x + b_2x \quad (4)$$

The alignment is set up by the ampersands in the two equations, in this case to make sure the equal signs line up. You can also use the `eqnarray` environment to take care of longer equations

```

\begin{eqnarray}
y = a_1x &+& b_1x + c_1x + d_1x \\
&\nonumber \\
&+& e_1x + f_1x \label{eq:y} \\
\end{eqnarray}

```

$$\begin{aligned}
 y = a_1x &+ b_1x + c_1x + d_1x \\
 &+ e_1x + f_1x
 \end{aligned} \tag{5}$$

In the examples you may have noted the larger spacing around the equal sign and addition symbol caused by the alignment. With AMS \LaTeX `indexAMS \LaTeX` (section 8.1), you can access another environment to do what we did with `eqnarray` above but which yields improved spacing. This is the `align` environment

```

\begin{align}
y = a_1x &+ b_1x + c_1x + d_1x \\
&\nonumber \\
&+ e_1x + f_1x \label{eq:y} \\
\end{align}

```

$$\begin{aligned}
 y = a_1x &+ b_1x + c_1x + d_1x \\
 &+ e_1x + f_1x
 \end{aligned} \tag{6}$$

In this case the alignment is much better. This is but one example of how using AMS \LaTeX improves your capabilities.

When you build equations, you need to know the commands that produce all the different mathematical symbols and notation. A good source to how to write mathematics is the guide by [Downes and Beeton \(2017\)](#). It is worth mentioning at this stage that while \LaTeX does excellent mathematical type setting, AMS has improved on the capabilities through their AMS \LaTeX extension. If you just need to type set a few formulas and ordinary equations, basic \LaTeX should be more than sufficient. But if you need some more special features, they are almost certainly available through the AMS \LaTeX extension.

Now turning to formulating an equation. When you type in an equation, you do generally not need to worry about spacing. As we noted earlier all letters will be typeset in mathematical italics, this is the norm for mathematical variables. The italics only applies to Latin letters, Greek letters are not italicised. This means that if you want to type in a function such as sine you need to be careful as this example shows

```

\begin{eqnarray}
\tau &=& \rho gh \sin\alpha \\
\tau &=& \rho gh \sin\alpha \\
\end{eqnarray}

```

$$\tau = \rho gh \sin \alpha \tag{7}$$

$$\tau = \rho gh \sin \alpha \tag{8}$$

In the upper row we typed in the letters ‘s i n’ and they came out in italics but with a similar spacing as between the variables g and h . In the second row we used the mathematical command `\sin`. In the first case, ‘sin’ is treated as three variable names s , i and n . \LaTeX , therefore has all the mathematical functions established as commands so that they are type set correctly. Adding letters after each other is just interpreted as a series of variables multiplied by each other. This has a bearing on those who persist to form variables more or less at abbreviations such as *ET* for evapotranspiration or worse *NDVI* for Normalized Difference Vegetation Index

$$NDVI = \frac{NIR - VIS}{NIR + VIS} \tag{9}$$

where NIR is the near infrared band and VIS is the visible. This clearly does not look good partly because of the way \LaTeX handles text in mathematical mode but also because it is a poor way to define variable names. So let this be a warning when you define your own variables and equations.

Despite the poor look of equation 9 we see that we can produce division. This is easily accomplished with the `\frac{}{}{}` command which takes two arguments. The first should con-

tain everything that belongs to the numerator and the second, everything that belongs to the denominator.

```
\begin{equation}\label{eq:NDVIb}
I_{\mathrm{NDV}} =
\frac{E_{\mathrm{NIR}}
- E_{\mathrm{VIS}}}{
E_{\mathrm{NIR}}
+ E_{\mathrm{VIS}}}
\end{equation}
```

$$I_{\mathrm{NDV}} = \frac{E_{\mathrm{NIR}} - E_{\mathrm{VIS}}}{E_{\mathrm{NIR}} + E_{\mathrm{VIS}}} \quad (10)$$

In this example we can see how the `\frac` command works but we can also see that it is possible to type in the equation in almost any form, in this case to try to make it structured, since it has no consequence for the output. You can also see a new command `\mathrm{}` which produces regular text inside mathematical mode.

When you use the `\frac` command and you need to put a parenthesis around the equation, you can obtain scalable parenthesis using the `\left` and `\right` commands

```
\begin{equation}\label{eq:NDVIb}
I_{\mathrm{NDV}} = \left(
\frac{E_{\mathrm{NIR}}
- E_{\mathrm{VIS}}}{
E_{\mathrm{NIR}}
+ E_{\mathrm{VIS}}}
\right)
\end{equation}
```

$$I_{\mathrm{NDV}} = \left(\frac{E_{\mathrm{NIR}} - E_{\mathrm{VIS}}}{E_{\mathrm{NIR}} + E_{\mathrm{VIS}}} \right) \quad (11)$$

The `\left` and `\right` commands must pair up. You can use `()`, `[]` and `{ }` and also `|` with these commands. It is also possible to set only one side of the pairs if you use a period for the side you want to omit: `\left.` `\right.`. Note that when you use the `\left` and `\right` commands, both must be included.

Making mathematical type setting is otherwise mostly finding the right commands to obtain what you want. There are many lists floating around that summarises these so I will not take up space here. I would argue that mathematics is one of the easiest things you can do in \LaTeX .

Try it

- Typeset some of your own equations or try to reproduce an equation out of an article or book, or both.

8 Extending the functionality of \LaTeX with packages

Because \LaTeX is Open Source, many enthusiasts contribute to \LaTeX functionality by providing extensions, so called *packages*. The [the Comprehensive \$\text{\TeX}\$ Archive Network \(CTAN\)](#) contains packages and their documentation and is thus a good source for information. I strongly recommend you to look at any package documentation posted on CTAN before using any package.

Out of all the available packages, only a small fraction will likely be of use to you. A well known effect of being new to \LaTeX is to go ‘package crazy’ and load almost anything. If this happens to you, then it will pass; but more severely, you may end up with unforeseen problems due to conflicts between certain packages. Some packages are also tailored to solve a particular problem for a given situation that may not apply to you. Much of this is stated in each package documentation.

In the following sections I will provide information on some packages that vary from a *must* to being of general interest. A package is loaded by adding the command `\usepackage` to the preamble:

```
\usepackage[]{} 
```

As can be seen the command has a normal argument field (denoted by the curly braces) and an optional input field (denoted by the square brackets). Many packages can be loaded without any options which means you can ignore the square brackets. Other packages require or has options for providing specific features. Such options are described in the package documentation.

I have ordered the packages alphabetically in the following.

8.1 amslatex

The `amslatex` package ([AMS LaTeX Project, 2020](#)) provides many improvements and extension to \LaTeX native mathematics capabilities. The package is loaded by

```
\usepackage{amslatex} 
```

There is no point to try to describe the package here but if you have extensive needs to write mathematical equations you should use the package. Carefully look at the [AMS \$\text{\LaTeX}\$ information at the American Mathematical Society](#). The package introduces ways to reproduce advanced mathematical constructions and should be used if your writing contains much mathematical notation.

8.2 appendix

The `appendix` package ([Wilson and Press, 2020](#)) adds functionality to appendices in an essay. \LaTeX has built in capabilities for handling appendices but there are short-comings that are remedied by this package.

The `appendix` package introduces the `appendix` environment which is preferable to the native `appendix` command in \LaTeX . When you write an appendix the heading should include the word `Appendix` and the label of the appendix, a capital letter. The same should appear in your table of contents.

The `appendix` package allows you to provide some options that provides the appropriate behaviour of appendices. The `toc` option places a header into the table of contents before listing the different appendices.

To implement the appendix you need to load the `appendix` package in the preamble by

```
\usepackage[titletoc]{appendix} 
```

This sets up the `appendix` environment to show the word ‘Appendix’ before the letter of the appendix in the table of contents using the option `titletoc`.

The appendices should reside within the `appendices` environment

```
1 \begin{appendices}
2   appendices appear as sections
3 \end{appendices} 
```

A problem that arises is that the numbering of tables and figures are not altered by the `appendices` environment. It is thus necessary to modify the figure and table counters so that they also show the letters of each appendix and also restart in each appendix. This can be accomplished by manually adding the following code after the section command in each appendix

```

1 \renewcommand{\thefigure}
2     {\thesection\arabic{figure}}\setcounter{figure}{0}
3 \renewcommand{\thetable}
4     {\thesection\arabic{table}}\setcounter{table}{0}

```

The code redefines the commands `\thefigure` and `\thetable` which determines the labels of figure and tables so that they reflect both the appendix ‘number’ A, B, C, etc. and the number of each. The command `\thesection` takes the definition of the section numbering and adds the arabic number of the figure or table so that you receive A1, A2 etc. The command `\setcounter` allows you to reset the numbers so that the numbering starts from one in each appendix for both figures and tables.

Note that the code given above will be required after the section command in each appendix.

8.3 babel

The `babel` package (Bezoz and Brahms, 2022) contains ability to switch many built in features that are language dependent to a language other than English. If you wish to write a document in Swedish you simply load the package as

```
\usepackage[swedish]{babel}
```

The document will then be shown with many small automatically generated differences to the English version. Certain headings such as ‘Contents’ and ‘References’ will be in your chosen language, dates will be written in the standard of your language, etc. Please refer to the [babel package documentation](#) for details.

With `babel` it is also possible to write multilingual documents and switch between languages mid-document. This is accomplished by first defining the languages of the document

```
\usepackage[swedish,UKenglish]{babel}
```

and then switching between the two using

```
\selectlanguage{swedish}
```

and

```
\selectlanguage{UKenglish}
```

depending on which language is locally activated.

The `babel` package is complex and it is necessary to read the documentation carefully to understand its capabilities to the fullest.

8.4 booktabs

The `booktabs` package (Fear, 2020) is a small package that improves table formatting. The package is loaded by

```
\usepackage{booktabs}
```

It is the `booktabs` package that allows you to use the commands `\toprule`, `\midrule`, `\bottomrule` and `\cmidrule` in tables. I have consistently used these for the tables in this text unless stated otherwise. The table on page 25 shows the use of these commands. The package provides a few other commands as well but the `booktabs` [package documentation](#) is well worth reading because it outlines ideas around good practises in table design.

Booktabs lines have been used throughout this compendium unless otherwise stated.

In addition to the booktabs package it is worth looking at the extension to creating tables using the package siunitx (section 8.18).

8.5 caption

The caption package (Sommerfeldt, 2022) provides easy tools for changing how the figure and table captions are formatted but also extends on the native functionality. The package is loaded by

```
\usepackage{caption}
```

In the standard layout the captions start with ‘Figure 1:’ and ‘Table 1:’ Since this part of the caption is created automatically, you cannot manually change the colon to a period, which is the format required by most journals, in the text. The following commands changes the formatting of the caption start:

```
\renewcommand{\figurename}{Figure.}  
\renewcommand{\tablename}{Table.}
```

Note that you can enter whatever formatting you want in the second argument. A perhaps quicker way is to load the package with options for fonts

```
\usepackage[labelsep=period,font={small,it},justification=justified]{caption}
```

With this you will receive a caption written in ‘small’ size italics font, and with the words Figure and Table followed by a period. This is shown in ‘Figure’ 4 (which is just a caption). Please refer to the caption [package documentation](#) for more details.

Figure 4. This is a trial caption to see how the different changes will look in the case of a caption in a float

8.6 dcolumn

The dcolumn package (Carlisle, 2014) provides means to define new alignments for tabular environments. The package is loaded by

```
\usepackage{dcolumn}
```

To provide an example of what can be accomplished with dcolumn we can define a new column type called ‘.’ (period) using the command \newcolumnntype

```
\newcolumnntype{.}{D{.}{.}{-1}}
```

This new column type will align numbers on the period. As you can see the command takes many arguments. The first argument is the name of the column type (‘.’). The second argument is divided into three parts, the first part indicates what character should be used for the alignment (the period), the second part shows what symbol should be used for the separator (usually the same as the first), the third part indicates how many decimal places should be shown. With -1 as the third argument, the column will be centred on the decimal point. An example

```

\begin{tabular}{. . .}
\toprule
45.73 & 43.894 & 5.463\\
33 & 0.0001 & 0.02\\
7.76 & .2 & 9.75\\
\bottomrule
\end{tabular}

```

45.73	43.894	5.463
33	0.0001	0.02
7.76	.2	9.75

And then the same table but with a different column type called ‘:’ (colon) which has the `\cdot` as decimal separator and 4 decimal places (the maximum number of decimals in the table; `\newcolumnntype{:}{D{.}{\cdot}{4}}`)

```

\begin{tabular}{: : :}
\toprule
45.73 & 43.894 & 5.463\\
33 & 0.0001 & 0.02\\
7.76 & .2 & 9.75\\
\bottomrule
\end{tabular}

```

45.73	43.894	5.463
33	0.0001	0.02
7.76	.2	9.75

As you realise you can do a lot with this package and define your own numeric column types. The `dcolumn` package documentation ([Carlisle, 2014](#)) is quite brief so experimentation is best way to learn this package.

The `siunitx` package (section 8.18) also has table alignment support which may be useful to study.

8.7 float

The `float` package ([Lingnau, 2001](#)) adds a placement option `H` to the `figure` and `table` environment (see section 6). The option is a stronger placement directive than the native `h!` option. The package is loaded by

```
\usepackage{float}
```

8.8 fontspec

The `fontspec` package ([Robertson, 2022](#)) is intended for expanding the use of fonts in \LaTeX . This package requires \LaTeX documents to be compiled under either \XeLaTeX or \LuaLaTeX . It is thus not possible to use \pdfLaTeX when using this package. Note that when using this package you may need to be well acquainted with font handling on your computer. When you use Overleaf you are dependent on fonts residing on Overleaf. You will always be able to include TrueType (`.ttf`) and OpenType (`.otf`) font files in your project in which case the font files need to be explicitly defined.

The package allows you to use system default fonts directly in \LaTeX . To include a font you use three commands which will set the main font (a serif font), a sans serif font and a monospaced font (such as Courier). These command are implemented as follows

```

1 \usepackage{fontspec}
2 \setmainfont{Times New Roman}
3 \setsansfont{Verdana}
4 \setmonofont{Courier}

```

The `fontspec` package description provides much more details on font usage with \XeLaTeX . As stated earlier, working with non-standard fonts may require a little work and be more involved than what I can cover here.

8.9 geometry

The geometry package (Umeki, 2020) provides an easier interaction with L^AT_EX page layout settings. The package is loaded by e.g.

```
\usepackage[a4paper]{geometry}
```

In this example, the option a4paper has been added in order for margins etc. to be scaled correctly for that paper size. This is the minimum of what you need to add to your preamble. It is worth noting that the documentclass command also takes a4paper as an argument but I prefer to use it in the call to geometry.

If you need to change your margins, you do this by simply using the following

```
\usepackage[a4paper,left=35mm,right=30mm,top=30mm,bottom=25mm]{geometry}
```

which just happens to be the call made for this document. This sets the left and right margins to 25 mm wide and the top and bottom margins to 30 mm. You should study the geometry package information (Umeki, 2020) for more details

8.10 graphicx

The graphicx package (Carlisle, 2021) is a *must*. Without this package you will not be able to include graphics into your document as we mentioned in section 5. The package is loaded by

```
\usepackage{graphicx}
```

in the preamble of your document.

When you have graphicx loaded into your document you have access to the command for including graphics, appropriately named \includegraphics. The command come with several optional arguments and looks, for example, as follows

```
\includegraphics[width=0.5\textwidth,angle=45]{fname.ext}
```

The file name is provided as the main argument in curly braces. It is not necessary to type the extension. It is also possible to provide a path to the graphics file if it is not in the folder where your L^AT_EX document resides. Valid graphics formats are JPEG, PNG, and PDF. Note that SVG and TIFF are not supported directly but can be included by converting the graphics to PDF. There is a package called SvG but it relies on having Inkscape as a backend to convert SVG code to a pdf file that is included in the L^AT_EX document. Please refer to the SVG package documentation (Ilten and Hanisch, 2020) for more details.

The optional arguments in the example provide L^AT_EX with the information that the figure should be reproduced with a width that is half the width of the text area. This type of relative scaling is convenient but you can also specify a fixed width in, say, millimetres, by writing, for example [width=57mm]. In addition we have also asked L^AT_EX to rotate the figure by 45°, which, of course, is unusual. In addition you can scale the figure by its height in a similar way to the width. There is a command \textheight that allows you to scale relative to the height of the text area.

If you provide only one of width= or height= the figure will be scaled proportionally in the other directions. You can scale an image dis-proportionally by providing different scaling factors for width and height.

In section 5 we also introduced the commands

```
\graphicspath{}
\DeclareGraphicsExtensions{}
```

that allow us to specify paths to graphic files as well as expected extensions of graphics files. If used, it will not be necessary to provide such information in the `\includegraphics` command. The commands need to be placed in the preamble of the document.

8.11 hyperref

The `hyperref` package (Rahtz et al., 2022) enables you to make workable links in your pdf output. The package is loaded by

```
\usepackage{hyperref}
```

When you use `hyperref` all the cross-links in your document becomes clickable and allows you to move around in the document. The table of contents is also clickable by default as are any references. In addition, you have access to the command `\href{}{}` which allow you to produce links to web URLs outside of your document. To show an example how this works we can set up a link to [TeX StackExchange](http://tex.stackexchange.com)

```
\href{http://tex.stackexchange.com}{\TeX\ StackExchange}
```

The command takes two arguments. The first is the complete URL and the second is the text that will be highlighted as a link in the text.

You can modify the way links are shown by the command `\hypersetup`

```
1 \hypersetup{
2   pdftoolbar=true,           % show Acrobat's toolbar?
3   pdfmenubar=true,          % show Acrobat's menu?
4   pdfwindow=fit,            % window fit to page when opened
5   pdfstartview={FitH},      % fits the height of the page to the window
6   pdftitle={title},         % title
7   pdfauthor={name},         % author
8   pdfsubject={subject},     % subject of the document
9   pdfkeywords={keyword1} {key2} {key3}, % list of keywords
10  pdfnewwindow=true,         % links in new window
11  colorlinks=true,           % false: boxed links; true: colored links
12  linkcolor=black,           % color of internal links
13  citecolor=SUBBlue,         % color of links to bibliography
14  filecolor=blue,            % color of file links
15  urlcolor=blue              % color of external links
16 }
```

Note that you do not need to list all settings, simply the ones you wish to change from default values. Just remember to separate them by a comma. With these settings, the links to external pages are blue, internal links are black except citations which are in a blue defined as `SUBBlue` (see section 8.22). As you can tell, the `hypersetup` also dictates how the pdf should be opened by Acrobat reader. You should refer to the `hyperref` package documentation (Rahtz et al., 2022) for more information.

Some of the settings in the `hyperref` package can be very useful to look at. First the `colorlinks` determines whether the links will be coloured or identified by a box. The latter is quite unsightly so setting this to `True` is recommended. The `linkcolor` will determine the colour of internal links in the document. This is probably best kept black since you will otherwise see all figure, table etc. references in some colour. The `citecolor` determines the colour of literature cites in the text. This can be set to a subtle colour to make in text references less obstructive to reading the text. In this text I have defined this as a blue colour. Many journals

use such colour schemes in the article files. The `filecolor` setting places links to files in a specific colour. Again, this may be advantageous to identify such links from other links in the document. Finally the `urlcolor` sets the colour of URL links in the document.

By selecting a few different colours in these settings you can achieve a document that conveys important linkages within and outside of your document.

8.12 inputenc

While scientific writing is primarily done in English and \LaTeX is made with English in mind it is often necessary to include words, for example, place names with accented letters in the text. Generic \LaTeX does, for example, not support the å, ä, and ö on the Swedish keyboard. You can of course create these letters through \LaTeX accenting commands, in this case by typing `\aa`, `\"a` and `\"o`, respectively. If this happens a few times it is not a major issue. There is however a way to ensure that what is on your keyboard is also possible to handle in \LaTeX .

The package `inputenc` (short for ‘input encoding’; [Jeffrey and Mittelbach, 2021](#)) allows \LaTeX to use an extended encoding, that is beyond the letters and numbers of the English keyboard. this is not limited to Swedish letters but to a large variety of other accented letters. The package should be called with the option `utf8`. The UTF8 encoding allows a computer to manage over a million different characters which includes all characters in modern Unicode type faces. By adding the following to your preamble, you have full use of your font character set. The package is loaded by

```
\usepackage[utf8]{inputenc}
```

Note that this package is not necessary when you use $\text{Xe}\text{\LaTeX}$ to compile your document.

8.13 lineno

The `lineno` package ([Böttcher and Lück, 2005](#)) does one thing, it allows you to add line numbers to, for example a manuscript. Some journals, for example, ask for such manuscripts. The package is loaded by

```
\usepackage{lineno}
```

You turn the line numbers on by inserting the command `\linenumbers` where you want the numbers to start. You can stop line numbers by placing the command `\nolinelnumbers` where you want them to stop. You should look at the `lineno` package documentation ([Böttcher and Lück, 2005](#)) for more details on different options.

8.14 lipsum

The `lipsum` package ([Happel, 2021](#)) produces a body of dummy text in your document. The text is commonly used in type setting to assess the look of the layout design and consists of a text body in improper Latin. By adding a number or range of numbers as an optional argument (1–4 in the example below) it is possible to display a specific number of paragraphs in your text. the benefit of this is that you can quickly add some text to see effects of your type setting design decisions without having to type anything into your document. The package is loaded by

```
\usepackage{lipsum}
```

and to add text to your document you add, for example,

```
\lipsum[1-4]
```

to where you want the text to appear.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

8.15 multirow

The multirow package ([van Oostrum et al., 2021](#)) allows writing table entries that span several table rows. The package is loaded by

```
\usepackage{multirow}
```

This package provides typesetting in the vertical similar to what is accomplished in the horizontal by the `\multicolumn{}{}{}` command provided by the `multicol` package. The basic command is `\multirow{number of rows}{column width}{text}` which take three arguments. The first determines how many rows the ‘multi-row’ entry should span. The second defines the column width of the ‘multi-row’ entry. The last is the text that should be displayed in the ‘multi-row’ entry.

The content of Table 1 was created using

```
\begin{tabular}{ll}
\toprule
\multirow{2}{30mm}{Common text a} & Column text 1\\
& Column text 2 \\
\midrule
\multirow{3}{30mm}{Common text b} & Column text 3\\
& Column text 4 \\
& Column text 5 \\
\midrule
\multirow{4}{30mm}{Common longer text example} & Column text 6\\
& Column text 7 \\
& Column text 8 \\
& Column text 9 \\
\bottomrule
\end{tabular}
```

As can be seen from the example the entry that should span multiple rows is entered on the first row of the set of lines it should span. The corresponding column of the following entries is left empty. The package allows for additional adjustments which you can study in the package documentation.

8.16 natbib

The natbib package ([Daly, 2010](#)) is a must if you need Harvard style author-year referencing system in your work. The package is loaded by

Table 1. An example of a table using multirow.

Common text a	Column text 1
	Column text 2
Common text b	Column text 3
	Column text 4
	Column text 5
Common longer text example	Column text 6
	Column text 7
	Column text 8
	Column text 9

```
\usepackage{natbib}
```

I will not dwell on natbib functionality here since we will go into much more detail when discussing referencing in section 9. What may be of more interest here is a section of code that I recommend you to use in conjunction with the natbib package. This code referred to as natbibspacing.sty

```

1 \newdimen\bibspacing
2 \setlength\bibspacing\z@
3 \renewenvironment{thebibliography}[1]{%
4 \bibfont\bibsection\parindent \z@\list
5 {\@biblabel{\arabic{NAT@ctr}}}{\@bibsetup{#1}}%
6 \setcounter{NAT@ctr}{0}}%
7 \ifNAT@openbib
8 \renewcommand\newblock{\par}
9 \else
10 \renewcommand\newblock{\hskip .11em \@plus.33em \@minus.07em}%
11 \fi
12 \sloppy\clubpenalty4000\widowpenalty4000
13 \sfcode\.=1000\relax
14 \let\citeN\cite \let\shortcite\cite
15 \let\citeasnoun\cite
16 \itemsep\bibspacing %
17 \parsep\z@skip %
18 }\def\@noitemerr{%
19 \PackageWarning{natbib}
20 {Empty `thebibliography' environment}}%
21 \endlist\vskip-\lastskip}
22 %-----
23 \setlength{\bibspacing}{0pt}
```

You may be able to load this code by adding the name to the call for natbib

```
\usepackage{natbib,natbibspacing}
```

but only if it exists in your L^AT_EX distribution. Otherwise, you can enter the code into your preamble after you have loaded natbib.

The natbibspacing is an example of pure code and may get an insight into how much details can be changed but also that it involves learning L^AT_EX as a programming language in detail. Fortunately, others in the community commonly have done this for you.

8.17 parskip

The default paragraph separation in \LaTeX is to use indented paragraphs. A second way to accomplish this is to use vertical white space between paragraphs. The package `parskip` (Mittelbach, 2021) allows for a simple way to change the paragraph separation.

The package is loaded along with any settings to be used using the following syntax

```
\uepackage{parskip}
```

This will implement default settings. Additional settings can be set in the package call. The package does not contain any new commands. The package is described in the `parskip` package documentation (Mittelbach, 2021).

8.18 siunitx

The `siunitx` package (Wright, 2022) is a very complex package that deals with how to write numbers and units (with focus on SI units). The package is loaded by

```
\usepackage{siunitx}
```

This package may to many seem a bit over the top but I argue that it will help to keep documents in good scientific order and is worth the effort. Let us look at some of the core ideas. Let us say we want to write a complex unit such as square volt cubic lumen per farad ($\text{V}^2 \text{lm}^3 \text{F}^{-1}$). This seems like a disaster waiting to happen. With the `siunitx` package and its command `\unit{}` we will get the correct units through the following

```
\unit{\square\volt\cubic\lumen\per\farad}
```

As you can see all the words we used to describe the unit in text is available as a command and the result is a perfect SI unit form. If you need to add a number to the unit there is a second command `\qty{}`

```
\qty{203}{\kilo\gram\metre\per\second}
```

which yields 203 kg m s^{-1} . Now why is this good? There are several benefits. First, it becomes quite clear what your unit is. Second, the spacing between numbers and units and between the units themselves are constant and of accurate length. It is true that you end up writing a lot but the benefits definitely outweighs the disadvantages.

The package also has specific commands for writing numbers. This can be useful to get proper spacing for 5 digit and larger values or proper exponential notation

```
\num{203166}  
\num{3.45d-4}  
\num{-e10}
```

will yield 203 166, 3.45×10^{-4} and -10^9 , respectively. Note that that the command introduces spacing as a delimiter for thousands, millions etc.

You can also write lists and ranges in several different ways using the commands `\numlist`, `\qtylist`, `\numrange`, and `\qtyrange`. So the following

```
\numlist{10;20;30} \\  
\qtylist{0.13;0.67;0.80}{\milli\metre} \\  
\numrange{10}{20} \\  
\qtyrange{0.13}{0.67}{\milli\metre}
```

will result in 10, 20 and 30, 0.13 mm, 0.67 mm and 0.80 mm, 10 to 20, 0.13 mm to 0.67 mm, respectively. By default the quantity commands add units after each number. This can be suppressed by adding the following options to the commands

```
\qtylist[list-units = single]{0.13;0.67;0.80}{\milli\metre} \\
\qtyrange[range-units = single]{0.13}{0.67}{\milli\metre}
```

which yields 0.13, 0.67 and 0.80 mm and 0.13 to 0.67 mm, which the preferred way to write lists or ranges.

The way numbers and units are presented can be modified when you load the package. This has the benefit that you do not need to go through and manually change anything if you need to change the format, you simply change the options when loading the package.

Some of the features shown above can be set as default when loading the package by adding optional argument. I usually load the package as follows:

```
\usepackage[separate-uncertainty=true,
             multi-part-units=single,
             product-units=single,
             list-units=single,
             range-units=single]{siunitx}
```

The options indicate that wherever I have lists, ranges, products or multi-part units, the units will only be written out once. This would replace having to add the same options every time we write any of these constructions. The option

The package also introduces a new alignment for the `tabular` environment. The new `S` alignment is more advanced than what is accomplished by the `dcolum` package because it can take into consideration material that should go in front of or after a number.

The package contains much more on representing data in text so it is worth the time to carefully study the extensive `siunitx` package documentation (Wright, 2022).

8.19 threeparttable

The `threeparttable` package (Arseneau, 2010) adds to the way you can typeset tables with captions and footnotes. Load the package using

```
\usepackage{threeparttable}
```

With the package loaded you have access to two new environments `threeparttable` and `tablenotes` as well as the `\tnote{}` command that should be entered into the table environment.

A table is thus built by using the `table` environment in which you start the `threeparttable` environment. Within this environment you add the `\caption` command followed by your table information in a `tabular` environment. If you need to add a note to entries in your table you use the `\tnote{}` command by entering a number in the square brackets. Once the `tabular` environment has ended you start the `tablenotes` environment which is a modified list environment and add your numbered notes as numbered items in the list. You finally end the table by ending the `tablenotes`, `threeparttable` and `table` environments. The following code modified from the package documentation (Arseneau, 2010) summarises the setup:

```
1 \begin{table}[ht]
2   \begin{threeparttable}[b]
3     \caption{...}\label{tab:...}
4     \begin{tabular}...% or {tabular*}
5       ...42\tnote{1}&.... ...
6     \end{tabular}
7     \begin{tablenotes}
```

```

8      \item [1] the first note ...
9      \end{tablenotes}
10     \end{threeparttable}
11     \end{table}

```

Table 2 shows a simple example of a `threeparttable`. Note that the table caption is adjusted to the width of the table. Note that all tables in this document has been typeset with the package.

Table 2. This is an example of a simple `threeparttable` with notes on specific entries

Mundane	Interesting	Scientific
Something ¹ Common	Something else Somewhat different ²	Sensational Very different

¹ the first note

² the second note

8.20 tikz/pgf

The `tikz/pgf` package (Tantau, 2021) is actually two packages. Tikz is a front end for the drawing capabilities set up by pgf. TikZ is a recursive name, *Tikz ist kein zeichenprogram*. The package is loaded by

```
\usepackage{tikz}
```

The `tikz` package documentation is 1321 pages (at the time of writing, version 3.1.10) and hence extremely detailed. The manual doubles as a tutorial and dictionary. We will briefly look at two examples that shows you what is needed and what can be done. It is noteworthy that this is still just scratching the surface of what can be done. Please visit [T_EXsample.net](https://www.texample.net) for numerous examples of Tikz output.

The first example shows a flow diagram for data collection at Tarfala Research Station.:

```

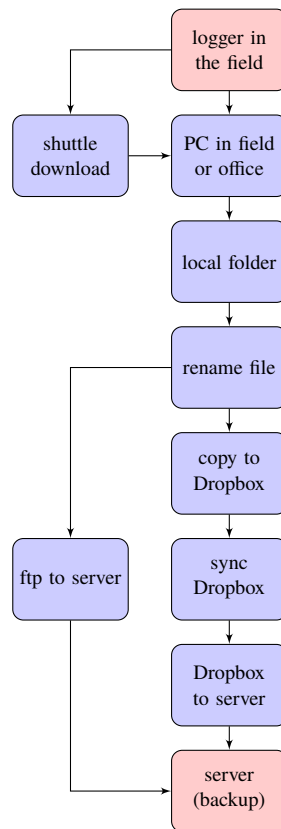
1  \tikzstyle{block} = [rectangle, draw, fill=blue!20,
2    text width=5em, text centered, rounded corners, minimum height=4em]
3  \tikzstyle{rblock} = [rectangle, draw, fill=red!20,
4    text width=5em, text centered, rounded corners, minimum height=4em]
5  \tikzstyle{line} = [draw, -latex']
6
7  \begin{tikzpicture}[node distance = 2cm, autoscale=0.7,
8    every node/.style={scale=0.7}]
9
10     \node [rblock] (logger) {\logger in the field};
11     \node [block, below of=logger] (pc) {\PC in field or office};
12     \node [block, below of=pc] (infolder) {\local folder};
13     \node [block, below of=infolder] (rename) {\rename file};
14     \node [block, below of=rename] (copy) {\copy to Dropbox};
15     \node [block, below of=copy] (sync) {\sync Dropbox};
16     \node [block, below of=sync] (toserver) {\Dropbox to server};
17     \node [rblock, below of=toserver] (server) {\server\\ (backup)};
18
19     \node [block, left of=pc, node distance=3cm] (shuttle)
20       {\shuttle download};
21     \path [line] (logger) -| (shuttle);
22     \path [line] (shuttle) -- (pc);
23
24     \node [block, left of=sync, node distance=3cm] (ftp) {\ftp to server};

```

```

25 \path [line] (rename) -| (ftp);
26 \path [line] (ftp) |- (server);
27
28 \path [line] (logger) -- (pc);
29 \path [line] (pc) -- (infolder);
30 \path [line] (infolder) -- (rename);
31 \path [line] (rename) -- (copy);
32 \path [line] (copy) -- (sync);
33 \path [line] (sync) -- (toserver);
34 \path [line] (toserver) -- (server);
35 \end{tikzpicture}

```



The second example is a figure I made for describing the energy fluxes at the ice surface on a glacier.

```

1 \begin{tikzpicture}[>=latex,scale=0.8, every node/.style={scale=0.8}]
2 \fill[top color=cyan!20,bottom color=black!30]
3 (0,-1) -- (0,0) -- (10,0) -- (10,-1);
4
5 \draw[thick] (0,0) -- (10,0); % Ground
6
7 \draw[<->] (5,-.9) -- (5,0); % Geothermal heat
8 \node at (5,-0.57) [right] { $G$ };
9
10 \draw[decorate,decoration=snake,segment length=11,->,blue!50!green]
11 (0,2.9) -- (0.8,0); % Incoming SW radiation
12 \node at (0.15,2.5) [right] { $I_S \downarrow$ };
13 \draw[dashed,decorate,decoration=snake,segment length=11,->,blue!50!green]
14 (0.8,0) -- (1.7,2.9);
15 \draw[dotted,decorate,decoration=snake,segment length=11,->,blue!50!green]
16 (1.2,0) -- (2,2.9); % Outgoing SW radiation

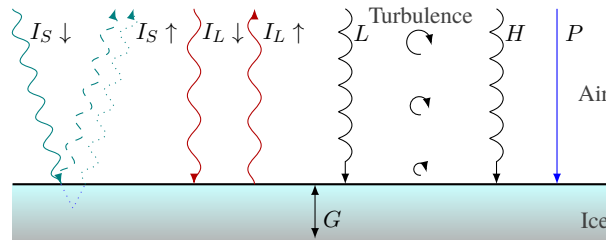
```



```

17 \node at (1.9,2.5) [right]  $\{I_S\uparrow\}$ ;
18 \draw[dotted,blue!70!white] (0.8,0) -- (1,-0.4) -- (1.2,0);
19
20 \draw[decorate,decoration=snake,segment length=18,->,red!70!black]
21 (3,2.9) -- (3,0); % Incoming LW radiation
22 \node at (3,2.5) [right]  $\{I_L\downarrow\}$ ;
23 \draw[decorate,decoration=snake,segment length=18,->,red!70!black]
24 (4,0) -- (4,2.9); % Outgoing LW radiation
25 \node at (4,2.5) [right]  $\{I_L\uparrow\}$ ;
26
27 \draw[decorate,decoration=coil,->] (5.5,2.9) -- (5.5,0); % Latent heat
28 \node at (5.5,2.5) [right]  $\{L\}$ ;
29
30 \draw [->] (6.75,.15) arc (280:0:.1cm) -- +(283:0.05cm); % Turbulence
31 \draw [->] (6.75,1.15) arc (280:0:.15cm) -- +(283:0.05cm);
32 \draw [->] (6.75,2.15) arc (280:0:.2cm) -- +(283:0.05cm);
33 \node at (6.75,2.8) [gray!50!black] {Turbulence};
34
35 \draw[decorate,decoration=coil,->] (8,2.9) -- (8,0); % Sensible heat
36 \node at (8,2.5) [right]  $\{H\}$ ;
37
38 \draw[->,blue] (9,2.9) -- (9,0); % Precipitation heat
39 \node at (9,2.5) [right]  $\{P\}$ ;
40
41 \node at (10,-.6) [left,gray!50!black] {Ice};
42 \node at (10,1.5) [left,gray!50!black] {Air};
43
44 \end{tikzpicture}
45

```



The examples above require certain tikz libraries to be loaded in the preamble. Which are required depends on what sort of graphic elements are required. The manual details this.

```

1 \usetikzlibrary{arrows,shadows,positioning}
2 \usepackage{pgfplots}
3 \usetikzlibrary{decorations.pathmorphing}
4 \usetikzlibrary{decorations.shapes}
5 \usetikzlibrary{patterns}

```

Although these examples may be daunting, they indicate the flexibility of \LaTeX .

The cover of this compendium was designed using tikz which shows that tikz can be used in a variety of ways.

Try it

- Visit the TExsample.net site and try out a few examples. If possible try to make changes to the code to familiarise yourself with the tikz programming.

8.21 titlesec

The `titlesec` package (Bezoz, 2021) provides tools for adjusting the formatting of title and section headings. The package is loaded by

```
\usepackage{titlesec}
```

As an example, the section titles in this report were set using the following command and settings:

```
\titleformat{\section}
{\normalfont\sffamily\Large\bfseries}
{\thesection}{1em}{}
```

The subsection and subsubsection titles can be adjusted using the same command. The package is, however, much more involved and will allow you to create more intricate formatting including page headers. Please refer to the [package documentation](#) for more details.




















8.22 xcolor

The `xcolor` package (Kern, 2021) introduces colour to L^AT_EX. In the `hyperref` and `tikz` packages we could see colour already and `xcolor` is in fact loaded already by those packages. The package is otherwise loaded by

```
\usepackage{xcolor}
```

With this package you can access a large number of predefined colours, the basic set is shown in Table 3. These are described in the [xcolor package documentation](#).

Table 3. The basic set of colours provided by the `xcolor` package. Note that the spelling follows US English.

Name	Colour	Name	Colour
red		lightgray	
green		brown	
blue		lime	
cyan		olive	
magenta		orange	
yellow		pink	
black		purple	
gray		teal	
white		violet	
darkgray			

In addition you can define your own colours using the `\definecolor` command. The command takes three arguments, first the name you wish to give to the colour, second the color space (cmyk, rgb etc.) and third the colour combination that makes up the colour. The following define the official Stockholm University colour palette in cmyk space

```
1 \definecolor{SUBBlue}{cmyk}{1.00,0.70,0.00,0.60}
2 \definecolor{SUOlive}{cmyk}{0.25,0.10,0.60,0.20}
3 \definecolor{SUSky}{cmyk}{0.35,0.00,0.10,0.00}
4 \definecolor{SUWater}{cmyk}{0.40,0.15,0.00,0.05}
```








```

5 \definecolor{SUFire}{cmyk}{0.00,0.65,1.00,0.00}
6 \definecolor{SUSilver}{cmyk}{0.12,0.08,0.08,0.23}
7 \definecolor{SUGold}{cmyk}{0.30,0.40,0.80,0.15}

```

Table 4 shows the official Stockholm University palette defined in CMYK through the combinations given on the SU web site (in swedish only).

Table 4. The official Stockholm University colours in CMYK and RGB. Note that the RGB colours do not include Olive, gold and silver since those should not be used on digital platforms. CMYK colours are used for printing.

Colour name SU	Colour \LaTeX		CMYK	RGB
Blue	SUBlue		1.00 0.70 0.00 0.60	0 47 95
Olive	SUOlive		0.25 0.10 0.60 0.20	–
Sky	SUSky		0.35 0.00 0.10 0.00	172 222 230
Water	SUWater		0.40 0.15 0.00 0.05	155 178 206
Fire	SUFire		0.00 0.65 1.00 0.00	235 113 37
Silver	SUSilver		0.12 0.08 0.08 0.23	–
Gold	SUGold		0.30 0.40 0.80 0.15	–

A useful web resource providing the RGB component define commands corresponding to colour swatches can be found at [\$\LaTeX\$ color site](#).

There are also ways to create hues out of the colours by mixing colours using the command `\color{}`. To make up a hue we specify how much of each to mix

```
\color{blue!40!red}
```

In the example the mix will be 40% blue and thus 60% red (since they must sum up to 100%) resulting in the purple colour of this sentence. The colours blue and red are part of the basic set of colours provided by the `xcolor` package (Table 3).

9 References

\LaTeX has the capability to handle literature references and generating reference lists in your documents. To accomplish this there are a few points we need to cover. First, we need to have the reference information available for \LaTeX to work with. This can be accomplished in two ways, with a data base containing the information or with reference information entered as text in the document. We also need to add functionality to the built in system for references in \LaTeX to be able to obtain author–year referencing.

9.1 Using a reference manager

The most efficient way to manage references in your research and thus when you write scientific reports is to use a reference manager. Such software allows you to build a data base of the literature you read and need for your texts. There are many reference managers available such as (alphabetically) [EndNote](#), [JabRef](#), [Mendeley](#), [PaperPile](#), [RefWorks](#), and [Zotero](#) to mention a few. The [Comparison of reference management software](#) page provides an overview of available tools. What you need to look for is that the tool can save or export the references in Bib \TeX format. The Bib \TeX format is a data base format developed for \LaTeX but has become one of the standards for reference information. You will see that most sites allow you to download

information in this format. The reference manager you chose must also be capable of generating a unique BibTeX ‘cite key’ (see below) for each entry. When you reference articles, you will do so by using the cite key in a similar way to what we saw how figures, tables and equations were tagged and referenced using the `\label` command. The different reference managers can provide useful bibliography files but it may involve additional steps. At the time of writing this Mendeley and Zotero directly integrate with Overleaf. Integration with PaperPile is in beta. RefWorks and JabRef will produce useful bibliography files, i.e. with proper cite keys, for inclusion in Overleaf files.

Unless you already work with a reference manager, I recommend the Open Source [Zotero](#) for reference handling. Any data base built in a specific manager can be migrated to other managers using standard reference data base formats so you will not be locked in to any software you decide to work with.

With Zotero you can save reference information from web pages and article pdfs directly into Zotero and quickly build your reference data base. Zotero works with a combination of storage on your own computer and a cloud based storage which is free up to 300 Mb. Start by signing up for a [Zotero](#) account and download and install the Zotero software according to instructions. You also need to install a connector for your web-browser (Chrome, Edge, Firefox or Safari). Finally, you should also install the [Better-BibTeX for Zotero](#) plugin in your desktop Zotero application. Just follow instructions to do so. Note that you can install Zotero on several computers and synchronise the data base between them. Synchronisation is accomplished through the cloud based copy of the data base which is updated and synchronised by Zotero whenever you add references to your data base.

Zotero allows you to also store pdf files with your references. This will, however, quickly fill your free data base space. I personally save only the reference information and not pdfs. At some point I had 2974 references in Zotero which used up 0.3% of my 300 Mb. So the free space will be sufficient for quite some time as long as you store only the reference information and not pdf files.

The main point here is that you need to start storing references in a reference data base software capable saving the data in BibTeX format for L^AT_EX. In order for L^AT_EX to handle references it is necessary for L^AT_EX to have access to the references in some ordered form, a data base file in BibTeX format is a key format.

9.2 Integration of Overleaf and Zotero

When using Overleaf you can link to your Zotero account. This is accomplished in the Overleaf account settings. Linking is also possible for Mendeley. Once you have completed this you will be able to transfer reference data directly from Zotero into Overleaf projects.

When you need to reference articles in your data base in a document you first need to obtain a copy of the data base in BibTeX (.bib) format to reside in your Overleaf project. When you press ‘Upload’ in your Overleaf project view to add new files, you will see several options in the pop-up Window that appears. One option will be ‘From Zotero’. This will copy your online Zotero data base into your Overleaf project. When you make the selection you will also have the possibility to name the file if the default name does not suit you. When you synchronise the data base from Zotero to Overleaf, the process may take a little while depending on the size of your data base.

When you add new information to Zotero, you also need to update the data base file in your project(s) to access these new references. The synchronisation between Zotero and Overleaf is not automatic. Updating is very easy to do. You simply click on the data base (.bib) file name in your project view. The data base file will be displayed in Overleaf and you have the possibility to ‘Refresh’ the data base using a button in the window showing the data base file. This transfers the updated version of the data base from Zotero to your project. You then have

access to any new items added to your data base. Note that each project has its own copy of the data base so you will need to update the data base file in all projects where you want the changes to be available.

When you work with the integration between Zotero and Overleaf, the data base will be provided Zotero default cite keys. These have the format *First author name–first word of the title–year of publication*. If you have set up Better BibTeX to generate a different cite key format in Zotero, that cite key format will revert to the default in the synchronisation process. At least at the time of writing this there is no way to work around this with the direct transfer.

If you instead want to work with your reference data base with your own Better-BibTeX cite key format, you need to save a copy of the data base to your computer from your desktop Zotero tool. Use the Export library function in Zotero and select ‘Better BibTeX’ format. You will then get a copy of the data base on your computer that you can manually upload to your project as a file. If you add new material to the data base you need to redo this procedure to get your data base updated in the project. This process is a little more time consuming than the synchronisation. If you have linked a Dropbox or git account to Overleaf you could save the updated data base file to those applications and synchronise the file that way. I will leave those options for you to investigate.

To work with your data base in your document to generate references and cross reference with a reference list, you will need a few additions to your document. The goal is to be able to relate any article you reference in your manuscript with an entry in your data base. We also want all references we added in the document to correspond to a reference in the reference list. We of course also want the reference list to correspond exactly to what we want to reference in the text. All this will be accomplished by L^AT_EX with the natbib package.

9.3 The natbib package

When you want to use *author–year* style referencing, the common style for the Earth Sciences, you need to use the natbib package (section 8.16). This package adds many new commands dealing with different types of references.

When you make references in the text you basically need two forms for the reference.

Active where the author name is part of the text and the year is within parenthesis as in ‘Beaty (1989)’

Passive where both the author name and the year are within parenthesis as in ‘(Beaty 1989)’

The natbib package provides two primary commands for this, `\citet{}` and `\citep{}`, respectively.

As with other cross-referencing the referencing works through labels also known as *cite keys* in the case of references. Each reference must therefore have a unique cite key. With Zotero to Overleaf synchronisation the cite key will by default be created from the first authors last name, the first ‘meaningful’ word in the title (thus ignoring words such as ‘The’ and ‘A’) and the year of publication. This default is what you will receive when you link and synchronise Zotero directly with Overleaf. With other reference managers the cite key may be created along different standards. So, a reference such as

Beaty CB, 1989. Great big boulders I have known. *Geology*, 17 (4), 349–352.

will by default be assigned the cite key `beaty_great_1989`. This means you can reference this article in the text as either `\citet{beaty_great_1989}` or `\citep{beaty_great_1989}` yielding ‘Beaty (1989)’ and ‘(Beaty 1989)’, respectively. Either one of these cite commands will enable L^AT_EX to add the reference to the reference list. Hence you simply add references and the reference list is built for you. Note that when you start typing the cite commands your

will get suggestions for what reference to add as soon as you type the first letter in the name. It is thus quite easy to add references and you do not need to memorise cite keys, just author names and possibly parts of the title of the paper.

The `natbib` package has several commands for referencing but the most commonly used are shown here

Command	Result
<code>\citet{}</code>	‘active’ citation ‘Beaty (1989)’
<code>\citep{}</code>	‘passive’ citation (Beaty, 1989)
<code>\citeauthor{}</code>	cites only author name, not year
<code>\citeyear{}</code>	cites only year, not author name

In all cases the Bib_{TEX} key must be entered as argument to the commands.

If you need to reference many authors in a passive reference, you simply add them all as argument separated by a comma. The cite command `\citep{beaty_great_1989, day_two_2014, knight_three_2015}` would yield (Smith 2013; Day 2014; Knight 2015). Note that they will not automatically be reorganised so it is important to enter them in the order we want according to the reference standard we use.

The `\citet{}` and `\citep{}` commands can also take optional arguments. If we want to have a reference that looks like (e.g., Beaty 1989; and references therein) we can write `\citep[e.g.,][and references therein]{beaty_great_1989}`. What we want added before the reference goes as the first optional argument and what should be placed after the reference goes in the second. If you do not want one or the other, you just leave the optional argument bracket empty.

If L_AT_EX discovers a Bib_{TEX} key in your document that does not exist in the data base file, you will see two question marks in the text: (??). This is how L_AT_EX signals errors where there is no match between the cite key in the text and keys in the data base file. It will also appear as a warning in your log file.

With the commands described above you should be able to add references to the text but in order for the system to work, you need to provide a source from which Bib_{TEX} can obtain the reference information and also information on how to format the references. For this we have two ways to go, the fully and the semi automated way. I will cover both but there is little point in making the semi automated system your standard so I include it for information.

9.4 The fully automated reference system

The fully automated system handles both the cross-referencing and builds the reference list for you. This means you are ensured to have a perfect match between what is referenced in the text and what appears in the reference list. In order to accomplish this you need to have several things in place. First you, of course, need to have your data base as a Bib_{TEX} `.bib` file in your project. You also need a file that tells Bib_{TEX} how to format your references. This is called a *Bib_{TEX} style file* (`.bst`). It is possible to custom make your own style but that is not something that is easily described here⁴. Instead you can browse [CTAN’s repository for bibliographic styles](#) and try them. Note that you should look for author-year formats. The repository contains many other formats as well. Many journal templates include their own style file. This means there is no single complete repository for all styles and some may simply not exist in `.bst` format. I would otherwise recommend going with the [Council of Science Editors \(CSE\)](#) style. It is a neutral international standard. Note that a style file is language specific (commonly English). If you need references in a different language a separate style file for that language must be created.

⁴Visit the [custombib](#) package page on CTAN if you want to try to create your own Bib_{TEX} style file

When you have your data base file and your style file, you can start using it in your document. First you need to load the `natbib` package in the preamble of your document.

```
\usepackage{natbib}
```

You then need to add the following code where you want your reference list to appear.

```
\bibliography{foo}  
\bibliographystyle{CSE}
```

The first command `\bibliography{}` tells Bib_T_EX which `.bib` file to look for, your bibliography file. The file name does not need to be written out with the extension `.bib` and you can add a path if your bibliography file is located in some sub-folder in your project. It is also possible to have more than one reference file from which to extract references, which can be useful when collaborating with other authors and each want to add ‘their’ references. However, with multiple `.bib` files there is a risk that the same cite key is duplicated between files which causes errors or warnings when the document is compiled. The second command tells Bib_T_EX which bibliographic style that should be used, the name of your `.bst` file here exemplified by the `CSE.bst` (Council of Science Editors). In this example we must thus have the files `foo.bib` and `CSE.bst` in our project.

With all this in place your reference list will be up to date whenever you add a new reference in the document and recompile it. Every reference in the text will correspond to an entry in the reference list and there will not be references in the list that are not referenced in the text. As mentioned earlier, a prerequisite is that the reference you reference must be in the data base file and all entries in the data base must have unique Bib_T_EX cite key.

Please remember that any references added in your data base software must be synchronised or uploaded to Overleaf before you can use the new reference.

If you are using the `hyperref` package you will encounter an oddity when you produce a table of contents. The table of contents entry for the reference list will not be tied to the heading of the reference list but to the last chapter in your report. The reason for this is that the heading ‘References’ is added by the `bibliography` command and is thus not recognised as a heading by `hyperref`.

In order for the reference list heading to be linked properly in the ToC you need to provide a command called `\phantomsection`. This places a marker on the page where the section header is located so that the `hyperref` can recognise its location properly and also for L^AT_EX to add the references header to the table of contents. So when you produce a reference list in a report and you are using `hyperref` you can use the following code to make sure the reference list is properly handled in your table of contents and that its hyperlink points at the correct page.

```
1 \phantomsection  
2 \addcontentsline{toc}{section}{\bibname}  
3 \bibliography{references}  
4 \bibliographystyle{CSE.bst}
```

9.5 The semi-automated reference system

The semi-automated reference system does not make use of a reference manager or an external bibliography file as discussed above. In the semi automated referencing system, L^AT_EX keeps track of the cross-referencing but you need to manually type in all references the way they should appear. First, you must have loaded the `natbib` package. Then you must add your references manually in an ‘`thebibliography`’ environment located where you want the reference list to appear. The environment with one reference added looks as follows.


```

1 \begin{thebibliography}{}
2
3 \bibitem[Beaty, 1989]{beaty_great_1989}
4 Beaty CB, 1989. Great big boulders I have known.
5 \textit{Geology}, 17 (4), 349--352.
6
7 \end{thebibliography}

```

Each reference must start with the command `\bibitem[]{}{}` which takes two arguments. the first argument is the in-text reference, the information you want \LaTeX /Bib \TeX to place in the text. Note that the comma is not a style, it is for \LaTeX to separate author and year. The second argument is the Bib \TeX key that is unique for all references. You need to manually create the unique cite key for all references. The key is what `natbib` uses to link a unique reference in the list with a `natbib` cite command containing the same key in the text. After the `bibitem` command follows the reference formatted the way it should look according to any author instructions that may be given for the reference list.

Using the system this way is very simple but has the draw back that while all references in the text will be matched by a reference in the reference list, there is nothing that prevents you from adding references in the list that do not appear in the text. This means the cross-referencing will always be correct but \LaTeX cannot do anything about the content in the reference list. You also have to manually type in the references in the desired format and check that each gets its unique cite key.

So to sum up, this way of using references in \LaTeX is useful to know but the fully automated system, described above, is what you should aim for.

10 The role of type setting

At this point it may be useful to try to outline why we should care about the look of our documents: the type setting of documents. An adage of type setting is that good type setting is never noticed. The role of type setting is to make the reading as effortless as possible. So understanding what constitutes good typesetting as well as what may disrupt good type setting, and thereby the ease of understanding a text is critical.

Type setting is a profession and requires knowledge. Before the personal computer type setting was accomplished by professionals. With the introduction of personal computers this task was placed in the lap of everyone who essentially lacked the knowledge to produce well formatted documents. To make it worse, the tools provided on the computers were not generated by professionals and to this day very little has improved in this sense. The result was an abysmal lowering of typesetting standards that we still live with many decades later.

The work by Donaldh Knuth to create the type-setting system \TeX in 1978 was the first serious attempt to create a computerised system for generating well formatted documents ([Knuth, 1984](#)). Despite almost predating the personal computer, \TeX and its simplified interface \LaTeX by Leslie Lamport ([Lamport, 1994](#), *Lamport \TeX . \LaTeX*) remains as a primary type setting tool.

Contenders for typesetting exist in the form of specific typesetting tools such as Adobe In-Design, QuarkXpress, and possibly Scribus. These are however designed for general typesetting and not the technical typesetting required in the sciences. Microsoft Word is not a typesetting software although many use it as such. It has severe drawbacks when considering true type-setting features.

Since type setting is a profession, it remains clear that a well type set document will involve some human intervention. Hence any computerised system to date will require some interaction from the user to overrule or tweak automatic settings imposed by the software. this will be true

for any automated system. Hence any person type setting a document should not expect every problem to be resolved automatically but expect to resolve final issues manually to obtain the best possible result.

Based on the previous statements, it becomes evident that we allow the software to do its best to resolve the typography and then we will interact with the software to resolve the final problems. In addition, it will be important to realise that any corrections we make in a document will primarily affect the remaining (1) paragraph, (2) section, and (3) document, in overarching order. Hence any document must be edited from the start to the end in order to not have edits negated by other edits made earlier in the document. This process may also have to be repeated.

In the following we will run through some typographical guide lines that are required to obtain a well formatted document.

10.1 Page formatting

When creating a document we need to consider the size of the text area and the size of the margins. It seems ‘normal’ to start with the margins and to let these determine the text area but in reality the working order should be the reversed.

A readable text require lines to not exceed a certain number of characters. There is no exact number but a span of characters that is considered readable. In short, the number of characters on a line should be between 50 and 75 characters. Shorter or longer lines will hamper reading and should be avoided.

The line length requirement is therefore the primary measure when designing a page layout. The text area must be limited to 50 to 75 character width. Try to avoid the extremes of the range. Based on the text width decision the remaining page width can be used for margins. Because different type faces have different character widths, the choice of type face and type face size will influence the optimal text area width.

Once the text area width has been decided there are two more decisions to be made. First the height of the text area must be decided and, second, the distribution of margins at the top, bottom, left and right of the text area must be decided.

The vertical size of the text area is to some extent limited by the amount of information that is required above and below the text area. As a minimum, and perhaps most commonly, the only information required is to have page numbers at the bottom of the page. In some cases it is required to have a page header containing chapter or other information. All such information requires some space that limits the text area vertically.

The distribution of margins is a second constriction to the text area. A simple format uses identical left and right hand margins. However, traditional book type settings use different left and right hand margins for left and right hand pages. The use of different margins for right and left hand pages only makes sense for printed matter and since most modern documents are produce for digital consumption, similar left and right hand margins are preferable.

Margins play a key role in readability. If the margins are narrow, the text appears to overwhelm the page. It is therefore better to keep the text area narrower rather than wider in order to simplify reading. However, remember that the size of the type face also plays a role and therefore the resulting line length. Since normal body text type face sizes are 10, 11, and 12 pt it is clear that 10 pt is small when typesetting an A4 size page. The user needs to make decisions on type face sizes so that line lengths and resulting margins obtain a good and readable balance.

At this point, it may be useful to point out that we try to aim for single column formats. In the past it has been common for, e.g., journals to use two-column format. The two column format may have benefits in that the normal type size is closer to 9 pt and hence more text can be squeezed into a page. In the digital world two column format is counter productive since a single page becomes harder to read on a computer screen. Hence we primarily aim for a single column format.

The two column-format have several other drawbacks. Since lines become very short, the reading gets interrupted by line breaks. It is also common that words must be hyphenated which also disturbs reading. Hence two (or more) column format is generally a poor typographical format and never a good choice for readability.

In conclusion the page layout should be designed based on (1) the type face size used in the text, enabling properly scaled line lengths and; (2) page margins scaled to adhere to requirements of footers and headers as well as left and right hand differences in type setting. So do when creating your page style, do not start with margins, start with the text area size.

10.2 Placement of figures and tables

Figures and tables. The placement of figures and tables in \LaTeX is determined by the float principles. The figure and table environments have default placement instructions but these can be over-ruled by adding specific suggestions for placement. The basic placement suggestions include ‘h’ for here, ‘t’ for top of the page and ‘b’ for bottom of the page. These can be combined to provide provisional placements and also augmented by ‘!’ for strengthening the command such as in ‘h!’ for ‘really here’. In addition there is a package called `float` which adds the placement ‘H’ (really here) which is stronger than the generic ‘h’.

[Mittelbach \(2014\)](#) provides a useful summary of the placement commands for floats.

Table 5. The placement specifier for tables and figure floats. The ‘H’ specifier requires the package ‘float’ to be loaded

Alignment	Description
!	indicates that some of the restrictions that normally apply should be ignored
h	indicates that the float is allowed to be placed in-line (‘here’)
t	indicates that the float is allowed to go into a top area
b	indicates that the float is allowed to go into a bottom area
p	indicates that the float is allowed to go on a float page or column area
H	a more strict ‘HERE’ placement command (Lingnau, 2001)

10.3 Final editing of your document

When you have completed your document and you need to submit it for publication you will need to carefully go through the document for final edits. Note that this only applies for documents that will be published from your original files and not edited further by a journal or book editor.

In a final manuscript you need to carefully look for typographical problems in your manuscript. Common problems to look out for can be summarised as follows.

- Are there so called orphans and widows in your document?
- Are there ‘rivers’ in your document from poor hyphenation?
- Are there poor line lengths due to improper hyphenation?

Let us look at each point listed above.

Widows and orphans. When you typeset documents it is not considered good formatting to have single lines spilling over from a paragraph from a previous page or to end a page with a line from a paragraph that continues on the next page.

A line that appears at the top of a page from a paragraph residing on a previous page is called an ‘widow’. A line at the bottom of a page that comes from a paragraph that continues

on a following page is called an ‘orphan’ (Figure 5). This terminology has its origins in the typographical guild. The main point is, however, that having single lines spill over to the next page or ending a page will disturb the continuity of reading and should be avoided at all costs.

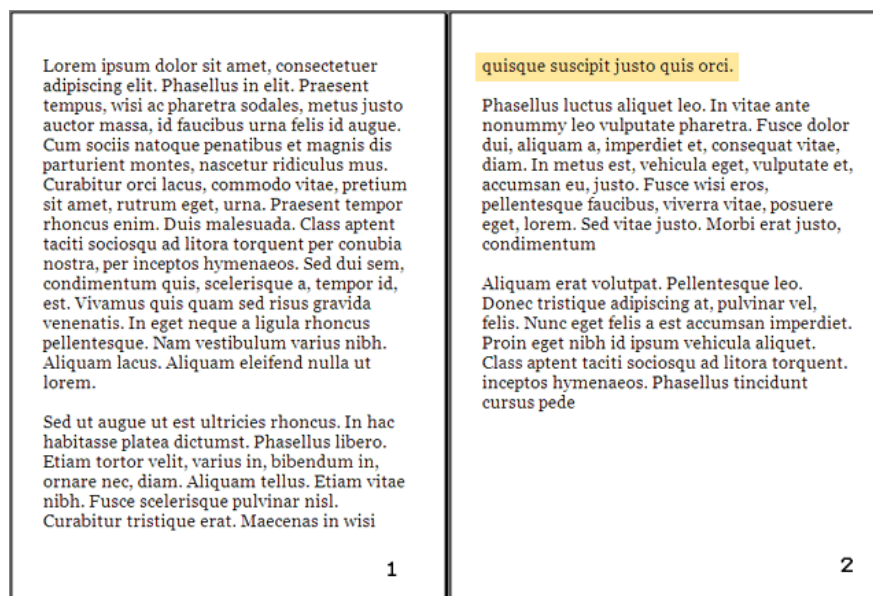


Figure 5. Example of an ‘orphan’ line of a paragraph spilling over to a following page. [Wikimedia](#)

There is no simple way to avoid orphans or widows. You can attempt to avoid these by several measures. One way is to change the text so that it becomes shortened or lengthened depending on what is necessary. Shortening is usually not the most problematic case. Note that this may involve shortening an earlier paragraph so as to remove a short last line of that paragraph to gain a line in the text.

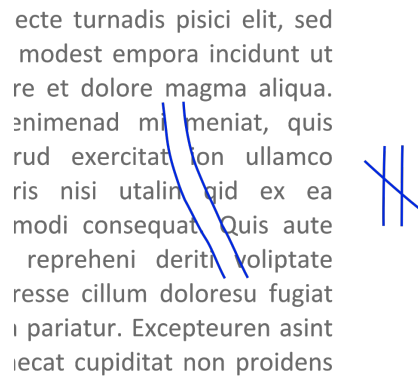
Lengthening the text is a more difficult remedy to avoiding orphans or widows. Any lengthening must involve adding valuable text that contributes to the value of the text as a whole. Expanding text is rarely the solution to solving issues of orphans and widows. An option for orphans is to simply break the page prematurely to make a new paragraph start on the following page. As a final note, I will add that the most required adjustments involve shortening of the text and not expansion, you need to carefully see how you can shorten your text to remedy the orphan and widow problem.

Adjustments may thus not only involve breaking the text but also considering rephrasing the text and performing deeper text editing to accomplish good paragraph breaks on the page. This can appear counter intuitive but remember that ease of reading is very important for understanding a text.

Rivers The notion of rivers in the text refers to the occurrence of white spaces in the text appearing as continuous areas of white space trickling through the text (Figure 6). This is an effect of poor hyphenation and is also quite common when line lengths are too short as in the case of two-column formats.

The best way to identify rivers

hyphenation A last measure for establishing good typography is hyphenation of the text. Figure 7 shows an example how three different software hyphenate the same text. The basis for hyphenation in \LaTeX is generic English by default. The `babel` package will provide more language specific hyphenation if you write in any other language than (US)English. However, regardless of primary language, scientific terminology will not be part of the default spelling and hyphenation dictionaries. Therefore, it will be up to the writer to manually go through the manuscript and suggest useful hyphenation of words that are causing Problems.



ecte turnadis pisici elit, sed
 modest empora incidunt ut
 re et dolore magma aliqua.
 animenad mimeniat, quis
 rud exercitaton ullamco
 ris nisi utaliquid ex ea
 modi consequat. Quis aute
 reprehendi derit voluptate
 resse cillum doloresu fugiat
 i pariatur. Excepteuren asint
 iecat cupiditat non proidsens

Figure 6. Example of a river in a block of text and how this is annotated in a print proof (blue notation).
 Source: [Wikimedia](#)

A typical error is the so-called ‘Overfull \hbox’ warning. This indicates that a line is overshooting its ideal length by some measure, the larger the value the worse. Many warnings are so small to be recognisable and can be ignored but it is always a judgement call.

To remedy errors concerning overfull ‘hboxes’, carefully go through your document and look for lines that overshoot your right hand margin. In many cases you will not notice issues in which case you can ignore them. However, in some cases you will see that words or hyphenations overshoot the line length and protrude into the margin which is a sign that you need to act. When this happens you need to take a look at the word that protrudes into the margin but not forget to see if there is a change in words or phrasing on lines preceding the problem that can remedy the problem. It is thus not just the line where the problem occurs that can help you fix the problem. The solution may require a larger scale resolution that affects an entire paragraph rather than a single sentence or line of text.

In conclusion, producing a good quality text involves working on the text so that it adheres to good typesetting rules. This involves not only formatting the text correctly but also adapting the text so as to make the text and the pages where the text occurs easy to read. Thus final touches always involve adjusting the formatting of the text in terms of both formulations and form of the text.

Typesetting a document is always an art that will in part be resolved by your software but always will require your manual touch and knowledge to be completed. Therefore, do not completely rely on your digital resource but always use your own knowledge to produce the best possible product of your work.

11 Automating \LaTeX or creating your own commands

Since \LaTeX is programmable you can accomplish almost anything. The problem is that it can become complicated. There is, however, a simple way to make life easier through the command `\newcommand`. This allows you to create your own command from the very simple to the very complicated. In its simplest form you can use the `\newcommand` to create a command that will replace something complicated to write with a short command name. Take the stable oxygen isotope ratio $\delta^{18}\text{O}$, for example. To typeset this you need to write

```
$\delta^{\text{\tiny{18}}}$O
```

So if you were writing a manuscript on stable isotopes, you may get really tired of repeating this over and over. We can then create a new command `\dO` that types this out for us

The first paragraph of Herman Melville's *Moby Dick* typeset using three different programs.
The text is set using Garamond Premier Pro 12/14 in a 5 cm wide column, fully justified.
Created by Roel Zinkstok of Zink Typography (www.zinktypografie.nl), January 2010

Microsoft Word 2008

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off – then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

Adobe InDesign CS4

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off – then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

pdf-L^AT_EX 3.1415926

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off – then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

Hyphenation and inter-word spacing statistics			
	Word	InDesign	pdf-L ^A T _E X
Number of hyphenations	9	10	4
SD of rws (pt)	2.26	1.94	1.42
Maximum rws (pt)	14.4	13.2	9.0
Number of lines with rws > 9 pt	5	2	0

SD: standard deviation; rws: inter-word spacing

Figure 7. Example of automatic hyphenation in a small width column between Word (2008), Adobe InDesign (CS4) and L^AT_EX (3.1415926). Lower number of hyphenations in a text are always preferable.
Source: Zinktypography

`\newcommand{\d0}{\delta^{18}0}`

The first argument to `\newcommand` shows the new command we want to create and the second argument shows the code that the new command will create.

When you create a new command, it is important not to use an existing command name. \LaTeX commands are always in lower case and \LaTeX is case sensitive. I therefore suggest you write yours with, for example, some capital letters, so called *camel case*. If you, for example, wanted to create a command called ‘do it now’ this would be written as `\DoItNow`, that is the first letter in each word is capitalised.

The `\newcommand` can be used with additional input. Let us look at the new command `\Nada` defined by

```
\newcommand{\Nada}[3]{\textsc{#1} \textit{#2}\ \textbf{#3}}
```

The first argument defines the command name `\Nada`. The second tells \LaTeX that there will be three arguments needed for the new command. The third argument shows \LaTeX what to do with the three arguments. In our case the first given by `#1` will be set as small caps, the second as italics and the third on a new line as bold. With input like

```
\Nada{What On Earth}{kind of joke typesetting}{is this?}
```

we get

WHAT ON EARTH *kind of joke typesetting*
is this?

Another example is to simplify an existing \LaTeX command. In section 4 we used the command `\multicolumn{}{}{}` to add table entries that could span several columns. To simplify this we can, for example, create

```
\newcommand{\MC}[3]{\multicolumn{#1}{#2}{#3}}
```

(MC for MultiColumn) and simply write `\MC{}{}{}` in the table instead.

Even though the first example is pointless in content the message is that you can automate repetitive and boring tasks. The `newcommand` can contain up to nine arguments. In fact the cover page of the compendium is produced by a command with five arguments used as follows

```
1 \Cover{Writing science with \LaTeX}%
2     {Peter Jansson}%
3     {\includegraphics[width=23cm]{LaTeX_text_cover.png}}%
4     {-1.5mm}%
5     {-1.5mm}
```

which means I can use the same command to generate a different compendium cover quite easily. The cover command consists of 27 lines of \LaTeX code and uses Tikz for placing the image and the text on the cover page.

There is also a related command `\renewcommand` that allows you to use an existing command name and redefine it. This can be very useful but also quite ‘dangerous’ if you do not know what you are doing.

Try it

- Create your own new command.

Try to replicate this using the `marginpar` (section 3.13) as a starting point

12 Copying documents from Word to L^AT_EX

Why have a section on moving Word content to L^AT_EX? By now you probably see there are some significant differences between how Word works and how L^AT_EX works. Most evident is the fact that while Word does all formatting in the hidden, most such work is upfront in L^AT_EX through commands. This can cause much grief when trying to copy text originally written and formatted in Word to L^AT_EX. Since you are likely to encounter persons who do not work with L^AT_EX, you may find yourself in a situation where most of a document is prepared in Word and left for you to, for example, transfer into a journal class template. Or, you may have an old favourite document you wish to move to L^AT_EX. In this section I will provide some hands-on guidelines for transferring material between the two tools as smoothly as possible.

Before continuing I need to mention that there are tools that translate Word files into L^AT_EX. The tools are usually good at replicating the look and feel of the Word document in L^AT_EX, but since that is rarely what we want, we end up with a code that mimics Word documents well but with a L^AT_EX code that is unnecessarily complicated. I recommend you to try such tools just to get a sense of what they can do but in the following we will work on the principle of preparing the word document so that the content can be copied using ‘copy–paste’ between Word and your L^AT_EX editor. Obviously there are also L^AT_EX to Word (or similar) translators but I will not discuss these further.

The first thing we need to realise is that no Word-based formatting will carry over into the L^AT_EX editor, only the text and the returns indicating end-of-paragraph. What order we make the following changes is immaterial, each find their own way. In short, anything that has required some form of formatting in Word needs to be addressed to work in L^AT_EX.

The basic strategy I recommend is to ‘L^AT_EXify’ your Word document already in Word before cutting and pasting it into the L^AT_EX editor.

Paragraph breaks L^AT_EX uses empty lines to identify paragraph breaks, Word does not. You need to go through and add empty lines between all paragraphs otherwise, you will otherwise have a hard time finding them in the L^AT_EX editor.

Italics and bold face The italics and bold face typesetting will not be carried over. It is much simpler to go through the Word document and look for your italics and bold text and use the commands `\textit{}` and `\textbf{}` to mark those words in Word than in your L^AT_EX editor where the formatting will be lost.

Super and subscripts The super and subscripts will not be carried through so format these with `\textsuperscript{}` and `\textsubscript{}` or if it concerns mathematical notation the `\^` and `_`.

Section headings The section headings formatting (levels) will not carry through to L^AT_EX (unless they are numbered so that you at least can identify them). This means it is easier to insert the commands `\section{}`, `\subsection{}`, and `\subsubsection{}` in the word file prior to copying.

Mathematics If you use equations and you have formatted variables with super and subscripts in the text, you should rewrite these using the mathematical mode already in Word.

Scientific units Mixing text and mathematical mode to produce super and subscripts when writing scientific units can be quite a mess. I strongly suggest you retype units using the commands in the `siunitx` package.

Dashes You need to replace en-dashes in Word with the `--` (double dash) format used by L^AT_EX.

Tables There is no easy way to prepare the tables for \LaTeX in Word. Just remember that any fancy formatting will be lost so it is best to keep the table in as simple form as possible. Complex tables are hard to typeset anywhere so this is where you probably need to spend some time.

A useful tool to try is `excel2latex`, a plugin for Excel that allows you to save \LaTeX code from a section of an Excel spreadsheet. If you take your tables from Word into Excel then this tool will yield a good basis for your table, quick and easy. At the time of writing the macro should work with Excel 2012–2016 (both 32 and 64-bit). In fact if you are uncertain about the coding of tables, this macro may be a good way to learn by making a table in Excel and then saving it for \LaTeX .

Figures Since you cannot copy graphics from Word into \LaTeX the only thing that will be copied is the figure caption. If you have your figures in Word and cannot produce originals for \LaTeX you can right-click on the figure in Word and save it as a picture. Use PNG format for line graphics and JPEG for photographs if you do so.

References If you have used Zotero or some other reference manager to produce references and reference list in your Word document then all such couplings disappear when you copy the text. Your references should, however be correct unless you decide to do some serious editing in \LaTeX after copying. You basically have two ways to deal with the references. One is to stick to the manual way and simply manually check for correct cross-referencing. The other is to load the `natbib` package and either enter/import references into `BibTeX` and use that system or go halfway and use the `\bibitem` and different `\cite{}` commands to produce the cross referencing. Either way this could cost you some time.

Extended type faces Depending on what system you use for compiling your \LaTeX document, extended type faces may be come an issue. This stems from the introduction of type faces that contain a much larger number of characters than used by older computers. $\text{Xe}\text{\LaTeX}$ uses these modern type faces and may compile a text containing such extended character sets correctly while $\text{pdf}\text{\LaTeX}$ generally cannot. Switching to $\text{Xe}\text{\LaTeX}$ may solve issues and for $\text{pdf}\text{\LaTeX}$ there are work-arounds (see the [pdf \$\text{\LaTeX}\$ documentation](#)). However, care should be taken when moving text containing, e.g. greek letters or mathematical signs to ensure they become type set correctly.

There may be additional details in the Word document that either do not translate well or are simply lost but the list above should normally cover 99% of a normal scientific document in Word.

13 Making presentation slides in \LaTeX

\LaTeX can be used for much more than creating regular documents. There are several different classes for creating presentation slides in \LaTeX . The most popular is called ‘Beamer’. Beamer is the German ‘term’ for a computer projector and has been so named by its German author.

Creating a simple Beamer presentation is not difficult. A single slide is made by the following code

```
1 \documentclass{beamer}
2 \begin{document}
3
4 \begin{frame}{frame title}
5   \((a=b)\)
6 \end{frame}
```

```

7
8 \end{document}\index{\BS frame}

```

What you see is that you need to make sure you use the beamer class. A slide is created by a new environment called frame. anything you place within the frame environment will be on your slide. Adding new slides is done by simply adding a new frame environment before or after the first.

There are of course many things you can add to your basic beamer presentation. The command `\frametitle{}` can be placed in a frame environment and will then produce title for that slide. However as shown in the example above you can add the title directly to the `\begin{frame}` command dispensing of the command itself. As with a regular document you can add a title page by providing the commands `\title{}` and `\author{}` in combination with `\titlepage` (note not `\maketitle`). A two page presentation with a title page can thus look as

```

1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}
6   \title{Presentation title}
7   \author{Your name}
8   \titlepage
9 \end{frame}
10
11 \begin{frame}{The title of the frame}
12   \((a=b)\)
13 \end{frame}
14
15 \end{document}

```

Beamer comes preloaded with a series of colourful layouts. These can be invoked by using the `\usetheme{}` and `\usecolortheme{}` commands. The themes and colour schemes have specific names which are described in detail at the [Beamer theme gallery](#). You can also create your won layouts but that requires a little bit of involvement.

In the end you should consult the [Beamer class user's guide](#) to learn more about all the possibilities available in Beamer.

14 Reinventing the wheel? ... or not

As with all programming languages, there will be bits and pieces of code that you will need over and over. You will soon find several such pieces as you develop your own 'style', good examples are all the packages that you will soon learn not to be able to live without. To prevent having to re-enter all this information in every new document you use, you can start to build up your own 'style' file.

L^AT_EX allows you to read in external files into the file you are authoring. This could literally be anything from a file containing some plain text to a file containing more or less a complete document. There are several commands for entering files such as `\input{}` and `\include{}`, where both commands take a file name as argument. These two commands are quite similar but differ in that the `include` command adds the content on a new pages whereas `input` simply adds the input text right where the command occurs. But a third possibility is to input a file of commands, to replace much of the preamble. This is accomplished with the now familiar `\usepackage{}` command and where your file should be named with the extension `.sty`.

What you need to do to start building your own style file is to add all the preamble information you think you will use repeatedly and unaltered into a file `fname.sty`. This file must not

contain the `\documentclass` and `\begin--\end{document}` commands and environments or any text. It can only contain material you would normally put in the preamble. If we assume you have a file you are working on and you take everything in the preamble and paste that into a file called `MyStyle.sty` then your bare document will look like the following

```
\documentclass{beamer}

\usepackage{MyStyle}

\begin{document}

\end{document}
```

The `\usepackage` command will then take your file and use its content in order to format the current document.

It is important to realise a couple of things here. First, you will always need to place a copy of your style file along with whatever document you are working on. \LaTeX will not keep track of it for you. Second, if you enter additional instructions in the preamble, you need to make sure they do not interfere with your style file. If you add something before the `\usepackage{MyStyle}`, you run the risk of having whatever you want to do over-ruled by the content in your style file and if you add something after the call the new material may over-rule something you have decided will be part of your style.

It is possible to add some additional safety to the style file so that it will only be used with appropriate versions of \LaTeX etc. We can start the style file by adding the following commands

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{MyStyle}[2016/04/01 my LaTeX style]
```

This tells \LaTeX that the latest format of \LaTeX ($\text{\LaTeX}2\epsilon$) is needed and that this is ‘my’ style file of such and such date. At the end of the file we can add

```
\endinput
```

which tells \LaTeX that the style file content is ended. This command means you can add more material to the file after the `\endinput` command but which will be ignored. This can be useful if you add material you are unsure you want to use but do not want to lose.

So a complete style file should look like the following example

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{MyStyle}[2016/04/01 my LaTeX style]
3
4 % your preamble material:
5
6 \endinput
```

It is possible to take the style file much further and essentially build your own class. This is beyond the scope of this compendium but the document [\$\text{\LaTeX}2\epsilon\$ for and package writers](#) provides much more in depth information on the subject.

15 How does it really work?

For those who wonder how the \LaTeX system really works, I will provide a brief overview of what goes on when a document is being typeset by the \LaTeX engine. First we should recognise that beneath \LaTeX is the original typesetting language \TeX created by Donald Knuth in 1977.

\LaTeX is essentially a, albeit huge, set of macros that facilitate type setting of documents of various kinds. The name \LaTeX is short for Lamport \TeX after Leslie Lamport who wrote the \LaTeX system.

In its most basic form a \LaTeX text document is compiled by feeding a \TeX compiler the document and the \LaTeX format macros to generate a so-called Device Independent file (DVI). This file can then be used by different drivers to show the document on screen or to print it on a printer or create a pdf. To respond to different needs variants of the \LaTeX compiler has been created. One such is $\text{pdf}\text{\LaTeX}$ which directly generates a pdf-file as output without creating an intermediate DVI file. $\text{pdf}\text{\LaTeX}$ is probably the most widely used way to generate documents in \LaTeX (Figure 8).

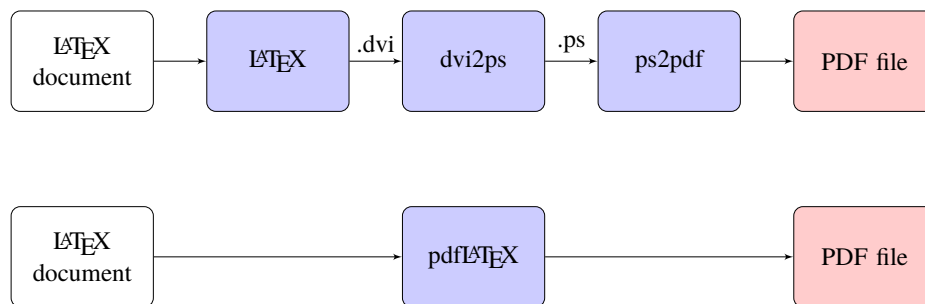


Figure 8. The basic way to generate a pdf file from a \LaTeX file. (*upper work flow*) Generic work flow running \LaTeX to generate a DVI file followed by running dvi2ps to produce a postscript file from the DVI and then ps2pdf to generate the pdf from the postscript file. (*lower work flow*) Work flow running $\text{pdf}\text{\LaTeX}$ to directly generate the pdf file.

Since \LaTeX in its generic form cannot easily handle system fonts a variant called $\text{Xe}\text{\LaTeX}$ has been created. $\text{Xe}\text{\LaTeX}$ can access system fonts in a more direct way but because there are internal differences between $\text{pdf}\text{\LaTeX}$ and $\text{Xe}\text{\LaTeX}$ we must be careful when trying to use certain packages in documents. Some (most) work with $\text{pdf}\text{\LaTeX}$ while some require $\text{Xe}\text{\LaTeX}$. This is the price paid by maintaining backwards compatibility. However, any problem is easily handled with some understanding of the two systems and by not overusing packages that we do not know anything about.

The way \LaTeX handles information that requires recursive treatment is to store results in external files (Table 6). This applies to, for example, the table of contents, figure and table references, reference citations. The first run of \LaTeX generates lists of all such cross-referencing information while \LaTeX type-sets the document. This way the location of each cross-referenced object becomes known. \LaTeX then needs a second run to find all references to the objects and replace these locations with the number of the figure or table or reference of the article. The result is that the document is compiled, not once, but in fact several times in order for all cross-referencing information to be properly located. Once this is accomplished you will find several files generated by this process that contains information from the process.

Table 6. Some of the more common files (extensions) generated when compiling a \LaTeX document. Note that some files are generated only when specific implementations are used.

Extension	Description
.aux	Generic information, mostly cross-referencing
.bib	Bib \TeX reference data base file
.bbl	b ibliography produced by Bib \TeX
.bst	Bib \TeX b ibliography style file
.blg	b ibliography (Bib \TeX) l og file
.cls	\LaTeX c lass file
.dvi	d evice independent file
.lof	l ist of figures
.log	complete compilation diagnostics reported by \LaTeX
.lot	l ist of tables
.out	hyperref PDF bookmarks
.sty	\LaTeX package file
.tex	\LaTeX or \TeX document file
.toc	t able of contents

Depending on what you include in your document \LaTeX will need to be run several times and maybe also involve other supplementary programs such as Bib \TeX . As an example, we can view the series of runs necessary in order to produce a table of contents and a reference list using the fully automated bibliography (Figure 9).

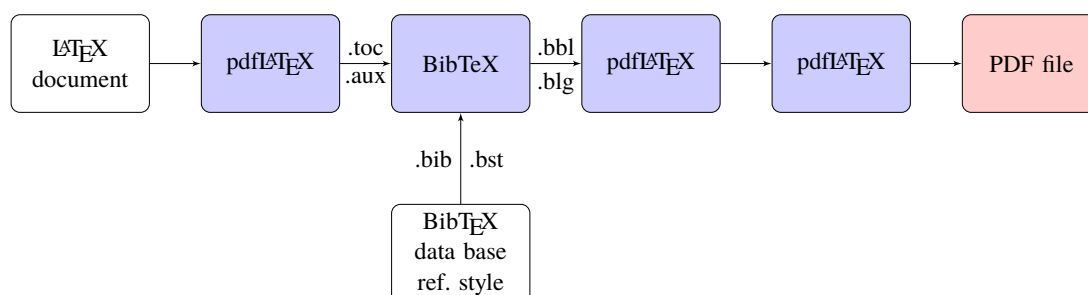


Figure 9. The pdf \LaTeX work flow for a document containing a table of contents (generating the .toc and a reference system based on a Bib \TeX data base. Note that pdf \LaTeX needs to be run twice after the Bib \TeX run in order for the cross referencing and reference list to be inserted correctly in the document. Many, if not all, \LaTeX editors take care of this work flow automatically.

16 Saving the worst to last

Errors. The one thing we all do not want to see. It is common to get errors in \LaTeX , some are simple to fix others may seem inexplicable. The one thing they have in common is that we have done something wrong. It is therefore useful to lay down a few ground rules to, first, reduce the numbers of errors and, second, to see how we can solve them. The best way to solve errors is to make sure they do not happen.

The first rule is thus, *be meticulous when you write your document*. The by far most common error comes from having forgotten to pair { and }. The second most common error is that you misspell a command. A third, concerns mathematical mode; you have either forgotten to switch back to text mode or you use a mathematical command in text mode. In all cases, the errors should be quite easy to spot and correct.

The second rule is to *write slowly* when you are doing more advanced L^AT_EXing. When you have some complex code you need to enter, do not rush and put a lot in at once. Instead complete the code bit by bit and make sure each bit works before you continue with the following parts. A useful tip is to maintain an empty document, I usually call `test.tex`, where I can build and test code before pasting it into the main document. This has two advantages, one is that you are working only on one piece of code in the test document and the other is that a small document compiles much faster than if you work in your main and most likely much longer document.

Sometimes, you will come across errors that may be ridiculously difficult to spot and to solve. Such errors can come about for a multitude of reasons but, remember, you entered something into the document which did not agree with L^AT_EX, directly or indirectly. Therefore, go back to what it was you just entered and try to figure out why this may have caused an error. Is something incompatible? Have you misunderstood how a certain command works? There is no easy answer as to how to solve these problems, you need to figure out what you did to cause it. If it is possible, try to remove the part in the document that appears to cause the problem and put it into a separate file (the test file) and see if you can replicate the error there.

Occasionally, you end up having a recurring problem that just does not seem to go away and you cannot find a single problem. One detail to remember is the set of files generated during a L^AT_EX run (Table 6). It sometimes happens that one of these files contain material that causes the problem. If you have run L^AT_EX and then made some changes that will also change the content of these files, there may be conflicting information left from earlier runs. It is therefore useful to make a clean compile of the document by first removing all the temporary or intermediate files. You need to be careful, however, since they all will have the same main file name and just differ in their extension. What you should remove are typically the `.aux`, `.bbl`, `.blg`, `.dvi`, `.lof`, `.lot`, `.log`, `.out`, and `.toc` files; if they exist. You should at all costs *avoid* to remove any `.tex`, `.bib`, `.bst`, `.cls` and `.sty` files, of course. Many editors can do this for you safely. In Overleaf this is called a `recompile`. If the problem persists after such a recompile, then you unfortunately still have a problem in your document.

With time, and as you have created enough errors to solve, you will improve your problem solving ability. This may seem as a cumbersome way, but it is the price we pay for the flexibility and power of L^AT_EX, and something quite familiar to those who do programming. As stated earlier, the best way to avoid errors is to not make them so write slowly and carefully, particularly when you enter commands, and compile your document frequently to that you see when an error appears.

Acknowledgements

I do not know where I came upon T_EX in the first place but my first forays were encouraged by Henrik Egnell, then a professor in mathematics and the Univ. of Minnesota and a close friend. I went on to produce my PhD thesis in plain T_EX, something I would not necessarily encourage anyone to follow up upon. Several years later I again started to look into T_EX in the form of L^AT_EX, why I do not recall, and have not looked back. Now I do all my serious work in L^AT_EX, particularly through Overleaf.

By teaching L^AT_EX to both Master's and PhD students in Physical Geography, I get valuable feedback that allows me to hopefully improve the compendium over time. The teaching also forces me to keep updated on developments in L^AT_EX. In this respect the TeX.StackExchange site and its members cannot be sufficiently credited. It shows the power of open source software and the communal power of many enthusiasts and professionals.

Appendices

A Running LaTeX on your own computer

This section outlines the basic installations necessary to run \LaTeX on your own computer. In order to run \LaTeX you need an implementation of \LaTeX , a *distribution*, and a \LaTeX -compatible editor.

A.1 \TeX

Which \LaTeX -distribution you chose depends on which operating system platform you are on. For Linux, you most likely already have \LaTeX already installed. For Mac and Windows, you definitely need to install it. There is one implementation call \TeX Live which can be installed on all platforms; on Macs using its Mac version MacTeX . The web page provides good instructions on how to download and install the distribution. For Windows there is also a distribution called MikTeX . I would recommend installing MikTeX on Windows platforms. MikTeX has utilities to easily update \LaTeX . It is also possible to create a portable \LaTeX -environment by installing on a memory stick.

A complete distribution of \LaTeX can be large (1–2 Gb). I would nevertheless recommend to make a full installation rather than a partial. The reason for this is that \LaTeX contains a huge number of so-called packages that simplifies tasks and provides new features to the basic functionality. If you install the entire distribution, you are sure to have them within reach if you need them. It is, however, also possible to make minimal installations and have the \LaTeX -implementation download packages as necessary (applies at least to MikTeX).

\LaTeX in its basic form is a command-line driven software. It is thus possible to use it in command line mode. This is not a very smooth way to use it and so a front-end editor is a useful complement.

A.2 \TeX editors

Since a \LaTeX file is a simple text file any editor can be used to write a \LaTeX document as long as you remember to save it as a plain text file with the extension `.tex`. Since \LaTeX contains many specially formatted commands, it is very handy to have a designated \LaTeX editor. Fortunately there are many such editors and most of them freeware. I can recommend two similar editors called \TeX studio and \TeX maker . These editors are implemented on all three major platforms. In fact, \TeX studio is a separate version of the original \TeX maker and they therefore have very similar characteristics. \TeX studio is completely open source while \TeX maker is free but maintained by a closed group of persons. The choice therefore largely depends on which version you like the most. \TeX studio has the advantage of being more customisable than \TeX maker . Another popular editor is \TeX nicCenter . The MikTeX distribution also contains its own simple editor which is installed along MikTeX . A good shareware editor is WinEdt . All editors should work with whatever implementation of \LaTeX you use. WinEdt is not exclusively intended for \LaTeX and can also be used for other programming languages. For Linux users, there are good ways to use the Emacs editor.

\TeX studio allows you to use spell checking (and thesaurus) from OpenOffice. You can download spell checking dictionaries for (currently) 101 languages (not that you need them) from the [OpenOffice extensions page](#). Simply search on the word ‘dictionary’ and the language you are looking for. Please note that spell checking files for British and US English as well as French and German come with the \TeX studio installation. The files you download will have the extension `.oxf` but are in fact Zip-files. You can therefore rename the file to `.zip` and

unzip the files into the 'dictionaries' folder in your T_EXstudio installation and, *presto*, you have a new language installed. Please check the T_EXstudio manual for details.

A.3 Reference management

L^AT_EX contains its own reference management system called BibT_EX. BibT_EX is a database system for references that consists of a standardised database format for reference material. Many other systems can read and export to BibT_EX, such as (alphabetically) [EndNote](#), [JabRef](#), [Mendeley](#), [Paperpile](#), [RefWorks](#) and [Zotero](#) as well as many data bases including Web of Science. To manage your database you will need some form of software. There are two ways forward, either you chose an on-line service such as Zotero or Mendeley, or you go for a local service installed on your computer.

The online services [Zotero](#) and [Mendeley](#) provide means to build reference databases by accessing material directly from the web and is probably the best choice if you have not begun building such a database with any particular software. These services can provide the necessary information to L^AT_EX to enable you to reference data from your data base directly in your document. The data is stored on-line but you also have the option of installing a standalone version that can be synchronised with the on-line version. This is of course most valuable if you wish or must to work off-line.

If you want to keep your reference database only on your computer I strongly recommend [JabRef](#). JabRef is a Java-based reference manager. This also means it works equally well on all platforms and it is free! JabRef installs through the downloaded installer. Since JabRef is Java-based, it will also require Java to be installed on the computer. In Windows, you will be prompted to download it if you try to start your already installed JabRef without Java. Java installs easily through an interactive installer.

A.4 Yet more software that might be of use

Converting documents from Word to L^AT_EX or *vice versa* is not difficult in principle but can be really tedious. There are some tools that can be helpful for such conversions. One of the most advanced is the [GrindEQ](#) suite of applications. GrindEQ is not free and whether or not you find it useful is a matter of preference. You should try it out and see what you think. My own experience is that it is useful to a point. The L^AT_EX created from a word file still requires a bit of editing to be usable. Most often manual editing is the best option since it gives you a chance to go through the document more carefully.

If you need to create tables, you can make use of a [Excel2LaTeX](#) Excel macro. This installs in Excel and allows you to generate table output for inclusion in L^AT_EX. Currently the macro is supported up to Office 2012–2016 so use in later versions will have to be tested.

B Install L^AT_EX on your computer

B.1 Windows

Installing L^AT_EX and the editor of your choice should be a breeze. In the case of MikT_EX and T_EXmaker you basically run automatic installers. MikT_EX requires you to download an installer (be sure to chose 32 or 64-bit depending on your system) from the web page and to run it. You can chose to either download L^AT_EX or to install it directly. Depending on how fast or steady your internet connection is, you could chose either way but I recommend to use the download option. If you chose this option, make sure you download a full version and also note where the installer saves the file. You will find it in a folder called MikT_EX 21.6 Setup (depending on which is the current version you download). Once the download is complete you open the folder containing

the downloaded files and start the installer in that folder (e.g. `basic-miktex-21.6-x64.exe`, again, depending on which is the current version). Running this installer will install \LaTeX .

The editors install without problems by just running the installers.

B.2 Mac

The simplest way to install \LaTeX on the Mac is via [MacTeX](#). The site provides the `MacTex.pkg` file to install everything needed for running and maintaining your \LaTeX installation. There is also an optional `MacTeXtras.zip` which provides some additional tools for editing etc. Once installed you will find a Preference Pane for the selection of your \TeX distribution (i.e. version) but you can ignore this unless you have an older installation on your Mac. The basic `MacTex.pkg` installer provides BibDesk, \LaTeX iT, \TeX Shop, \TeX Works and Excalibur as well \TeX Live Utility. The \TeX Live Utility is used for updating and installing new packages (functions or tools) and can be found under `/Applications/TeX/TeX Live Utility.app`.

The other tools installed are useful but there are alternative applications available that may be more to your liking. `bbedit` is a very useful, freeware text editor that can parse \TeX using a [set of scripts](#). Alternatively, Latexian (available through the App Store) is a very clean, 'Mac-like' program that provides live previews of the compiled text as well as a good navigation interface.

B.3 Linux

A basic \TeX -environment is included in most Linux distributions.

C Learning more on \LaTeX

As with all high quality and hence popular freeware, the internet is a very good source for more information. There are several sources for information that deserves mention: The [\$\LaTeX\$ Project home](#) (see additional links on the site), the [\$\TeX\$ StackExchange](#), and perhaps the [\$\LaTeX\$ Community](#). The \TeX StackExchange is highly recommended because you can easily search for older asked questions, pose new ones (if searching the site did not yield an answer) and take part in discussions. In fact if you do a search with the keyword 'latex' and something you are looking for, StackExchange entries will inevitably be high on the search.

In addition to the web-based sources above, the Comprehensive \TeX Archive Network (CTAN) repository contains both files and manuals for thousands of packages that solves specific tasks in \LaTeX . The two main distributions Mik \TeX and \TeX Live both use CTAN as their source for packages so if you have installed a complete set-up of these you also have most of what is on CTAN installed and ready to use.

Getting to know the details of \LaTeX may be a little difficult because of the wealth of material available on the internet. There are, however a few books that may be of use such as [Mittelbach et al. \(2008\)](#), [Voss \(2010\)](#), and [van Dongen \(2012\)](#). [Mittelbach et al. \(2008\)](#) is a comprehensive book about the core features and packages of \LaTeX . It works well both for learning but also to be used as a dictionary. [Voss \(2010\)](#) is also mostly a reference handbook of different commands. It does not provide as much explanations as [Mittelbach et al. \(2008\)](#) but is well organized and much more condensed. [van Dongen \(2012\)](#) includes discussion of a very useful subset of packages in \LaTeX . It is a very good start and overview that is hard to find elsewhere but does not go into depth. An additional book worth mentioning is [Voss \(2011\)](#) which is devoted to typesetting tables. One book that also deserves mentioning but for a different reason is [Kottwitz \(2011\)](#). This book is aimed to be an introduction to \LaTeX and will only work as such. It is not very useful as a reference after its introductory values. You will also find many introductions to \LaTeX on the web, the most used is probably the so-called `lshort`, 'The Not So Short Introduction to \LaTeX '. There is also a [Wiki book on \$\LaTeX\$](#) which is under development.

... and then there is this document.

References

- AMS LaTeX Project. 2020. User's guide for the amsmath package (Version 2.1). LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/amslatex>.
- Arseneau D. 2010. The threeparttable package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/threeparttable>.
- Bezons J. 2021. The titlesec, titleps and titletoc Packages. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/titlesec>.
- Bezons J, Braams J. 2022. Babel. Localization and internationalization. Unicode TEX, pdfTEX, LuaTEX, XeTEX. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/babel>.
- Böttcher SI, Lück U. 2005. A LaTeX package to attach line numbers to paragraphs. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/lineno>.
- Carlisle D. 2014. The dcolumn package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/dcolumn>.
- Carlisle D. 2021. Packages in the 'graphics' bundle. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/graphics>.
- Daly PW. 2010. Natural sciences citation and references (Author-year and numerical schemes). LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/natbib>.
- Downes M, Beeton B. 2017. Short math guide for LaTeX. Technical report. American Mathematical Society.
- Fear S. 2020. Publication quality tables in LaTeX. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/booktabs>.
- Gregorio E. 2016. The package imakeidx. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/imakeidx>.
- Happel P. 2021. Access to 150 paragraphs of Lorem Ipsum dummy text. LaTeX package documentation. Comprehensive TeX Archive Network.
- Ilten P, Hanisch. 2020. The packages svg and svg-extract. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/svg>.
- Jeffrey A, Mittelbach F. 2021. inputenc.sty. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/inputenc>.
- Kern U. 2021. Extending the LaTeX's color facilities: the xcolor package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/xcolor>.
- Knuth DE. 1984. The TeXbook. Boston: Addison-Wesley Professional.
- Kottwitz S. 2011. LaTeX beginner's guide. Birmingham: Packt Publishing.
- Lamport L. 1994. LaTeX: A document preparation system, User's guide and reference manual. Second edition. Boston: Addison-Wesley Professional.
- Lingnau A. 2001. An improved environment for floats. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/float>.
- Mittelbach F. 2014. How to influence the position of float environments like figure and table in LaTeX? TUGboat 35:248–254.
- Mittelbach F. 2021. The parskip package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/parskip>.
- Mittelbach F, Goossens M, Braams J, Carlisle D, Rowley C. 2008. The LaTeX companion. New York: Addison-Wesley.
- Rahtz S, Oberdiek H, The LaTeX3 Project. 2022. Hypertext marks in LaTeX: a manual for hyperref. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/hyperref>.
- Robertson W. 2022. The fontspec package. Font selection for XeLaTeX and LuaLaTeX. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/fontspec>.
- Sommerfeldt A. 2022. Customizing captions of floating environments. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/caption>.
- Tantau T. 2021. The TikZ and PGF packages. Manual for version 3.1.9a. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/pgf>.
- Umeki H. 2020. The geometry package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/geometry>.
- van Dongen M. 2012. LaTeX and friends. Berlin: Springer.
- van Oostrum P, Bache , Leichter J. 2021. The multirow, bigstrut and bigdelim packages. LaTeX package

- documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/multirow>.
- Voss H. 2010. LaTeX quick reference. Cambridge: UIT Cambridge.
- Voss H. 2011. Typesetting tables with LaTeX. Cambridge: UIT Cambridge.
- Wilson P, Press H. 2020. The appendix package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/appendix>.
- Wright J. 2022. siunitx - A comprehensive (SI) units package. LaTeX package documentation. Comprehensive TeX Archive Network. <https://ctan.org/pkg/siunitx>.

Index

* option, 18
-out file, 64
-toc file, 64
.aux file, 63, 64
.bbl file, 63, 64
.bib file, 63, 64
.blg file, 63, 64
.bst file, 63, 64
.cls file, 63, 64
.dvi file, 63, 64
.lof file, 63, 64
.log file, 63, 64
.lot file, 63, 64
.out file, 63
.oxt file, 65
.sty file, 63, 64
.tex file, 63–65
.toc, 19
.toc file, 63
.zip file, 65
\$, 11
%, 12
&, 11
\-, 17
\DeclareGraphicsExtensions, 26, 36
\Huge, 15
\LARGE, 15
\Large, 15
\NeedsTeXFormat, 61
\ProvidesPackage, 61
\addcontentsline, 19, 50
\appendix, 18
\arabic, 32
\author, 9
\begin–\end environment, 13
\bibitem, 51, 59
\bibliography, 50
\bibliographystyle, 50
\bottomrule, 24, 32
\caption, 27
\cdot, 34
\chapter, 18
\cite, 22, 23
\citeauthor, 49
\citep, 48, 49
\citet, 48, 49
\citeyear, 49
\clearpage, 28
\cmidrule, 24, 32
\color, 46
\copyright (©), 16
\date, 9
\day, 23
\definecolor, 45
\documentclass, 8, 13
\endinput, 61
\eqref, 22
\footnote, 23
\footnotesize, 15
\frac, 29
\frametitle, 60
\frenchspacing, 18
\frontmatter, 18
\graphicspath, 26, 36
\hfil, 22
\hline, 11, 24
\href, 36
\huge, 15
\hypersetup, 36
\imakeidx, 20
\include, 60
\includegraphics, 26, 35
\indent, 17
\index, 20
\input, 60
\item, 11, 22
\label, 22, 27
\large, 15
\ldots (...), 16
\left, 30
\linebreak, 17
\linenumbers, 37
\listoffigures, 19
\mainmatter, 18
\maketitle, 9
\marginpar, 23, 24
\mathrm, 30
\midrule, 24, 32
\month, 23
\multicolumn, 25, 57
\newcolumnntype, 33
\newcommand, 55, 57
\newline, 17
\newpage, 17
\noindent, 17
\nolinebreak, 17

- `\nolinenumbers`, 37
- `\nonfrenchspacing`, 18
- `\normalsize`, 15
- `\num`, 40
- `\pagebreak`, 17
- `\pageref`, 22
- `\paragraph`, 18
- `\part`, 18
- `\phantomsection`, 20, 50
- `\printindex`, 20
- `\qty`, 40
- `\ref`, 22
- `\renewcommand`, 33, 57
- `\right`, 30
- `\scriptsize`, 15
- `\section`, 9, 18, 58
- `\selectlanguage`, 32
- `\setcounter`, 19, 32
- `\sin`, 29
- `\small`, 15
- `\subparagraph`, 18
- `\subsection`, 18, 58
- `\subsubsection`, 18, 58
- `\tableofcontents`, 19
- `\textbf`, 15, 58
- `\textheight`, 35
- `\textit`, 15, 58
- `\textregistered` (®), 16
- `\textsf`, 15
- `\textsl`, 15
- `\textsubscript`, 58
- `\textsuperscript`, 58
- `\texttt`, 15
- `\textwidth`, 22
- `\the`, 23
- `\thefigure`, 32
- `\thesection`, 32
- `\thetable`, 32
- `\tiny`, 15
- `\title`, 9, 14
- `\tnote`, 41
- `\today`, 23
- `\toprule`, 24, 32
- `\underline`, 15
- `\unit`, 40
- `\usecolortheme`, 60
- `\usepackage`, 31, 60
- `\usetheme`, 60
- `\usetikzlibrary`, 44
- `\year`, 23

- `\#`, 14
- `\$`, 14
- `\%`, 14
- `\&`, 14
- `\`, 17
- `\,`, 14
- `_`, 14
- `\,`, 14
- `LATEX`, 6
- `LATEXify`, 58
- `LATEXit`, 67
- `LATEX Font Catalogue`, 16
- `TEX`, 51, 61
- `TEXLive Utility`, 67
- `TEXWorks`, 67
- `TEXmaker`, 65, 66
- `TEXnicCenter`, 65
- `TEXshop`, 67
- `TEXstudio`, 65
- `TEX StackExchange`, 5, 7, 67
- `^`, 12
- `_`, 11
- align environment, 29
- amslatex package, **31**
- appendices environment, 31
- appendix environment, 31
- appendix package, 31
- article class, 8, 14
- babel package, 32
- bbedit, 67
- beamer class, 59
- Better-BibT_EX, 47
- BibT_EX, 46, 47, 66
- BibT_EX style, 49
- BibDesk, 67
- body text, 52
- bold, 15
- book class, 14
- booktabs, 24
- booktabs package, 32
- BS titlepage, 60
- bst file, 49
- camel case, 57
- caption package, 33
- case sensitive, 8
- center environment, 21
- cite key, 48
- class

- article, 8, 14
 - beamer, 59
 - book, 14
 - letter, 14
 - report, 14
 - slides, 14
- command with argument, 13
- command without argument, 12
- comment, 14
- commenting text, 12
- Comprehensive T_EX Archive Network, 5, 67
- conditional hyphen, 17
- Council of Science Editors reference style, 49
- cross referencing, 22
- CSE reference style, 49
- CTAN, 5, 30, 67
- curly braces, 10
- dcolumn package, 33
- description environment, 21, 22
- device independent file, 62
- display mathematics, 28
- document environment, 9, 13
- documentclass command, 13
- Donald Knuth, 61
- dvi file, 62
- dvi2ps, 62
- em-dash (—), 16
- en-dash (–), 16, 58
- EndNote, 46, 66
- enumerate environment, 21
- environment
 - appendices, **31**
 - appendix, **31**
 - center, 21
 - description, 21, 22
 - document, 9, 13
 - enumerate, 21
 - equation, 11, 13
 - flushleft, 21
 - flushright, 21
 - itemize, 13, 21
 - minipage, 21
 - quotation, 21
 - quote, 21
 - tablenotes, 41
 - tabular, 11
 - threeparttable, 41
 - verbatim, 21
 - verse, 21
- environment command, 13
- eqnarray environment, 28
- equation environment, 11, 13, 28
- errors, 63
- Excalibur, 67
- Excel2LaTeX.xla, 25, 59, 66
- figure environment, 27
- float, 27
- float package, 34
- float placement, 28
- flushleft environment, 21
- flushright environment, 21
- fontspec package, 34
- formatting
 - page, 52
- geometry package, 35
- graphicx package, 26, 35
- GrindEQ, 66
- grouping, 14
- heading
 - numbered, 18
 - unnumbered, 18
- hyperref package, 36, 45, 50
- imakeidx package, 20
- index, **20**
- inputenc package, 10, 37
- italics, 15
- itemize environment, 13, 21
- itemize list, 10
- JabRef, 46, 66
- JPEG, 35, 59
- Knuth, Donald, 4
- LaT_EX, 51
- Lamport, Leslie, 4, 62
- Latexian, 67
- letter class, 14
- line length, 52
- lineno package, 37
- lipsum package, 28, 37
- list
 - itemize, 10
- listoftables, 19
- lists, 10

- macro parameter, 14
- MacTeX.pkg, 67
- MacTeXtras, 67
- margins, 52
- math minus, 17
- math subscript, 14
- math superscript, 14
- mathematical mode, 14
- mathematics, 28
- MaxTeX, 67
- Mendeley, 46, 47, 66
- MikTeX, 65, 66
- minipage environment, 21
- multirow package, 38

- natbib package, 38, 48–50, 59
- natbibspacing.sty, 39
- non-breaking space, 14

- OpenType font, 34
- optional argument, 15
- orphan, 53, 54
- Overleaf, 7

- package, 30
 - amslatex, **31**
 - babel, **32**
 - booktabs, 24, **32**
 - caption, **33**
 - dcolumn, **33**
 - float, **34**
 - fontspec, **34**
 - geometry, **35**
 - graphicx, 26, **35**
 - hyperref, **36**
 - imakeidx, 20
 - inputenc, 10, **37**
 - lineno, **37**
 - lipsum, **37**
 - multirow, **38**
 - natbib, **38**, 48, 59
 - package, **31**
 - parskip, **40**
 - pgf, **42**, 42
 - siunitx, 33, 34, **40**
 - SVG, 27, 35
 - threeparttable, **41**
 - tikz, **42**, 42
 - xcolor, **45**
- page formatting, 52
- PaperPile, 46, 66

- paragraph break, 10
- parskip package, 40
- PDF, 35
- pdfL^AT_EX, 16, 59, 62, 63
- pgf package, **42**
- PNG, 35, 59
- preamble, 9, 13
- ps2pdf, 62

- quotation environment, 21
- quotation marks, 16
- quote environment, 21

- recompile, 64
- Refworks, 46, 66
- report class, 14
- reserved characters, 14

- sans serif text, 15
- secnumdepth, 19
- siunitx package, 33, 34, 40, 58
- slanted text, 15
- slides class, 14
- small caps, 15
- square brackets, 10
- starred command, 18
- starred option, 18
- sty file, 60
- style file, 60
- SVG, 35
- SVG package, 27

- tab stop, 14
- table, 24
- table of contents, 19
- tablenotes environment, 41
- tabular environment, 11, 24, 27, 41
- text area, 52
- thebibliography environment, 50, 51
- threeparttable environment, 41
- threeparttable package, 41
- TIFF, 35
- tikz library, 44
- tikz package, **42**, 45
- titletoc, 31
- tocdepth, 19
- TrueType font, 34
- two-column format, 52
- type face style, 15
- type setting, 51
- type size, 15
- typewriter text, 15

underlined text, 15

usepackage, 61

utf8, 10, 37

verbatim environment, 21

verse environment, 21

widow, 53

WinEdt, 65

Word to L^AT_EX, 58

WYSIWYG, 8

xcolor color set, 45

xcolor package, 45

XeL^AT_EX, 16, 59, 62

Zotero, 46, 47, 59, 66

Revision history

- v.7. 29 Sep., 2023** Added new section on general typographical aspects of layout. Added the packages `multirow` and `titlesec` to the list of recommended packages. The text on the `siunitx` package has been expanded and updated to follow version 4 changes of the package. Expanded section on Zotero to include new features in synchronisation with Overleaf.
- v.6. 10 Oct., 2022** Added details on adding an index and appendix. Also added an index to the compendium. Added references to the most recent package documentation on CTAN. Sections concerning Overleaf and Zotero has been updated to reflect current features.
- v.5. 9 Oct., 2021** The section on referencing has been partially rewritten and reorganised. Minor corrections throughout the text.
- v.4. 1 Oct., 2020** Revisions made to follow developments in Overleaf. Reorganisation in the section of package descriptions. Added a brief chapter on graphics in addition to presenting the `graphicx` package. General updates of links and typos.
- v.3. 15 Oct., 2019** The part on referencing is completely rewritten with notes on Zotero and Overleaf 2 integration included. Many minor typos and errors in v.2 have also been corrected throughout the document.
- v.2. 19 Sep., 2018** Added appendix on how to install \LaTeX on a personal computer. Minor typos in v.1 corrected. Adjusted for Overleaf v2.
- v.1. 29 Apr., 2016** First public version