Using LaTeX in Linguistics: A Handbook

Nina Markl & Brandon Papineau

16th April 2021

Contents

Chapter 1

Before you start

1.1 Installing TeXlive or MacTeX

Before you can start using LaTeX on your computer, you will have to install TeXlive (for Linux or Windows) or MacTeX (for Mac). These are comprehensive systems which include the most important programmes, macro-packages, fonts etc. that LaTeXneeds to run. Both distributions are free to download. Note that they are quite big and download and install might take a while.

1.2 Installing and setting up TeXstudio

In principle LaTeX can be used without a specialised editor. However, editors make it much easier to use LaTeX by allowing you to preview your document, integrate several documents, auto-complete commands, close brackets (which is very important) and generally help you along the way (try hovering over any command). There are many editors out there [1], including some which are web-based (Overleaf), and some which have an interface similar to word processors (Lyx) but the one we recommend is TeXstudio, which is freely available to download for Linux, Mac and Windows. Note that you can't use a word-processor like Word or Apple Pages to edit .tex files.

Once you have installed TeXstudio, make sure that the default *engine* is XeLaTeX. There are several different *engines* which are used to compile your file. The default engine is called pdflatex, but it is sensible to change this to XeLaTeX as it's more versatile as it natively supports unicode input. To change the default engine, go to Options > Configure TeXstudio... > Build and set the *default compiler* to XeLaTeX.

[1] `https://en.wikipedia.org/wiki/Comparison_of_TeX_editors`

Chapter 2

Introduction

2.1 What is LaTeX?

/ˈlɛɪtɛx/ or /ˈlɛɪtɛk/ is a typesetting system based on the programming language TeX. First developed by computer scientist/mathematician Donald Knuth in the 1970s, TeX was supposed to improve typesetting for mathematics in particular. LaTeX was developed by computer scientist Leslie Lamport (hence La-TeX) and is essentially package of macros built on top of TeX which automates a lot of the formatting tasks and makes the language a lot easier to use for writers (as you don't have to program everything yourself). The newest version of LaTeX is LaTeX 2_ε.

Unlike word processors (such as MS Word or Apple Pages), LaTeX is not "What you see is what you get". While you type your text you cannot see how the finished document will look. All editing is done in a plain text file with the extension .tex. If you want to include images or tables or create new sections or subsections you do so with specific commands, which are implemented once the document is *compiled* at which point you can also see your document as a PDF (and depending on your settings TeXstudio will provide you with a new .pdf file each time you compile your file).

2.2 Why use LaTeX?

There are many advantages to using LaTeX once you've got the hang of it. Since it handles all formatting for you (following the commands you provide), it allows you to focus on content rather than looks. When you insert tables or images LaTeX will try to find the best place for them in your document (though there are commands to override these settings). It is also very easy to include foot- and endnotes, references and citations, indexes, as well as sections and subsections, all of which are automatically numbered and can very easily be referred to (via labels). Bibliographies can be generated automatically (provided a file with the relevant information). Most

importantly perhaps for linguists (and other scientists), there are many packages in LaTeX dedicated to typesetting equations, syntax trees, phonological rules, and basically anything else you could ever need. It is also very easy to use different kinds of fonts (including IPA) and languages (even within one document).

Overall, LaTeX makes it easy to manage large (and small) documents, include any "non-standard" symbols, generate bibliographies and in-text citations and produce very professional looking documents.

Chapter 3

Basics

3.1 Setting up a new file in TeXstudio

First, open TeXstudio and create a new document. Give it a name and save it to a new directory. Every time LaTeX compiles your file, it creates a bunch of temporary and ancillary files alongside a PDF. All ancillary files TeXstudio creates will also be saved in this directory. It is in general easiest if all files that you want to include in your LaTeX file (images, bibliography files etc.) also are in this directory. This directory can be synced via the cloud using a service like OneDrive or Dropbox.

3.2 Compiling your file

As noted above, you cannot see changes you are making to your document in the editor while you are making them. In order to implement all the changes you are making, you need to compile the file. In TeXstudio, you can click the green arrow in the top bar (or press F6) to compile and the green double arrow (or press F5) to "build and view". You will need to "build and view" your document the first time you are compiling – all changes need to be implemented from scratch and all commands need to be run. Once you have done this once, however, you can click compile or F6 to implement changes you made. The difference is basically that if you "build and view" LaTeX runs everything from scratch, while compile simply runs the commands that have been added or changed (but not everything that stayed the same). In general, it's advisable to compile your document frequently because that makes it a lot easier to find the source of an error between document versions. (Though TeXstudio helpfully provides the line number of errors).

3.3 Preambles

Each document in LaTeX requires a preamble. The preamble is where you define the commands you will use in the main document. While you can manually define new or redefine existing commands, most commands are defined within the *packages* and the *document class* you load in the preamble.
This is part of the preamble for this document. We will address the different components of the preamble in turn:

```
\documentclass[a4paper, 12pt]{report}
\usepackage[british]{babel}
\usepackage{setspace}
\onehalfspacing
```

3.3.1 Document Class

The *document class* defines the type of document you are about to start. The classic document classes include `article` (for short documents such as academic articles), `report` (for longer documents with several chapters), `book` (for actual books), `proc` (proceedings), `minimal`, `beamer` (presentations). Different document classes have different labelled "levels" – articles have sections, subsections, subsubsections, paragraphs, while reports have all those and chapters. All levels are automatically numbered (though that can be suppressed by placing an asterisk before the curly brackets and the title). In addition to these standard document classes, there are also many specialised document classes, documentation for which can be usually be found on CTAN. We really like the KOMA script document classes, because they allow for a lot of customisation with respect to fonts, margins, spacing etc..

```
\documentclass[options]{class}
\documentclass[a4paper, 11pt]{article}
```

Once you have completed your preamble, you need to provide a command to start the document:

```
\begin{document}
your entire document goes here
\end{document}
```

3.3.2 Packages

As mentioned above, packages essentially define the commands you will use. If you load a package in the preamble, you can use all the commands it defines. If you have downloaded TeXLive you likely already have all packages you will use downloaded, and only need to load them by typing them in your preamble:

```
\usepackage{package}
```

Some useful packages will be outlined in chapter 6.

3.4 Commands

3.4.1 Basic Structure

Commands in LaTeX follow the structure:

```
\command
```

In commands that allow option these options are defined in square brackets before the main command:

```
\command[optional-parameter]{parameter}
\usepackage[german]{babel}
```

3.5 Some idiosyncrasies

3.5.1 Comments

Comments can be added using %. This symbol can also be used to comment out a line or entire section (Ctrl + T will comment out the selected section, Crtl + U will un-comment it).

3.5.2 Symbols

The following symbols either have a special meaning or are not available in all font and therefore won't print if you type them without any further markup:

```
% $ & {} ^ \
```

If you want to print them you will need to preface them with a backslash (\), or in the case of the backslash:

```
\textbackslash
```

3.5.3 Quotes

Single and double quotes are printed slightly differently in LaTeX than they are in word processors. Make sure to use right-facing single quotes (next to the "1" key) (1 for single quotes and 2 for double quotes) and the apostrophe for closing quotes (one for single quotes or two for double quotes).

Single quotes	`` ` ' ``	‘ ’
Double quote	`` `` '' ``	“ ”

Note that quotes (and hyphenation as well as section numbering and bibliographies) are language specific. You can use a language support package like `babel` or `polyglossia` to define which language(s) you will use in the document and to adjust everything accordingly, using a second package called `csquotes`. In the case of `babel`, you simply add `\usepackage{babel}` to your preamble and add any languages you want to use to the optional parameter. If you type "british" in the optional parameter of this package (`\usepackage[british]{babel}`), all dates printed in the document will follow European (rather than default American) conventions. Note that you can also insert several languages but that LaTeX will use the first language for things like the "References", language-specific quotation and date conventions (e.g. `\usepackage[german, british]{babel}` will print "Bibliographie"). `polyglossia` is an alternative to `babel` specifically for XeLaTeX. It requires the package `fontspec` and allows you to set a default language

(\setdefaultlanguage{language}), main language (\setmainlanguage{language}) and other language(s) (\setotherlanguage{language1, language2, language3}) in the preamble.

3.5.4 Line breaks and spaces

LaTeX does not use white space (blank, tab) in the same way that word processors do. Any amount of white space is treated as one unit.

```
Look        at this    white space
```
Look at this white space

Similarly, hitting enter will not give you a linebreak:

```
There
are
no
linebreaks
here.
```
There are no linebreaks here.

Overall, LaTeX takes care of the optics and layout of your document – so it's best not to worry too much about line- and page breaks. However, if you really want to start a new line or new page you can tell LaTeX to do so:

```
This is a linebreak \newline
now there are two\newline \newline
linebreaks.
```

This is a linebreak
now there are two

linebreaks.

You can also start a new paragraph with indentation:

```
This great paragraph ends here.\par
Here's a new paragraph.
```

This great paragraph ends here.

Here's a new paragraph.

A page break is inserted with the command `\pagebreak` (but it's best to only use this if sparingly and only if really necessary).

To insert more white space, use the command `\vspace{Xcm}`.

To change the linespacing, you can use the package `setspace` (see also 6) which has the predefined commands `\onehalfspacing` and `\doublespacing`.

3.5.5 Fonts

As already noted XeTEX is a TEX typesetting engine which uses Unicode and also supports the use of any fonts you may have already installed on your machine (e.g. all the fonts used in a word processor like MS Word). While LATEX comes with an extensive set of fonts you can use pretty much any existing font you like and tinker with many of them to adjust them to your purposes using the package `fontspec`.

The fact that XeTEX further works with unicode input also means that a lot of what used to be annoying about LATEX is no longer an issue. If you use an compiler like `pdflatex` (dating from pre-unicode times) any "special" characters (including accents, Umlauts, IPA symbols, etc.) need to be encoded with a special command.

`fontspec`

The main package regarding fonts used in conjunction with XeLaTeX is `fontspec`. Use it without any further optional parameters to make the Unicode input work. To set your own fonts you can use the following commands, which will define different fonts for different contexts (headings, body etc.).

```
\setmainfont{font}[font features]
\setsansfont{font}[font features]
\setmonofont{font}[font features]
```

You can select any fonts you have installed. Note that the second parameter is optional. If you want to use IPA symbols, one way to do this is to use unicode

fonts with IPA support such as Charis SIL, Cambria, Gentium, Junicode or Brill. (There is another way to typeset IPA via the package `tipa` which we will also cover in section 6.3.1).

Font features

While the fonts are defined in the preamble, single sections of text can be individually formatted.

One nice feature of LaTeX is that you can just tell it to *emphasise* text without having to specify (or worry about) how it is emphasised (italics, underlined etc.). LaTeX will choose a style depending on context:

```
Here is some \emph{emphasis on these important words}.
```

Here is some *emphasis on these important words*.

```
\textit{Here is some \emph{emphasis on these important words}}.
```

Here is some emphasis on these important words.

You can, of course, also choose font features for short section. (But once again, it's generally a good idea to leave the formatting details to LaTeX).

Italics (Ctrl+I)	`\textit{Italics}`	*Italics*
Small capitals	`\textsc{Small capitals}`	Small capitals
Bold (Ctrl+B)	`\textbf{Bold text}`	**Bold text**
Slanted	`\textsl{Slanted text}`	*Slanted text*
Underlined	`\underline{Underlined}`	Underlined
Typewriter	`\texttt{Typewriter}`	`Typewriter`
Sans serif	`\textsf{Sans serif}`	Sans serif
Colour[1]	`\textcolor{red}{Red}`	Red
Strikethrough[2]	`\st{Strikethrough}`	~~Strikethrough~~

[1] requires the package `xcolor`
[2] requires package `soul`

You can also change font sizes – this can be done globally (i.e. for the whole document) in the optional parameter of the document class or more locally:

Tiny	`\tiny{tiny}`	tiny
Scriptsize	`\scriptsize{scriptsize}`	
Footnotesize	`\small{small}`	scriptsize
Small	`\normalsize{normal}`	
Normal	`\large{large}`	small
Large	`\Large{larger}`	
Larger	`\LARGE{very large}`	normal
Very Large	`\huge{huge}`	
Huge		large
		larger
		very large
		huge

3.6 Mathmode

Sometimes you will need to use `mathmode` - most obviously when you want to typeset mathematical equations (using the package `amsmath`), but also sometimes when typesetting primarily mathematical symbols. The `mathmode` environment is defined by $:

```
$>$
```

$>$

An equation using `amsmath` looks like this[3]:

```
\begin{align*}
\frac{\partial u}{\partial t} = u^2w - Bu + C\left(\int_{-\infty}^
\infty \phi(x-y)u(y,t)\operatorname{d} y - u(x,t)\right)
\end{align*}
```

$$\frac{\partial u}{\partial t} = u^2w - Bu + C\left(\int_{-\infty}^{\infty} \phi(x-y)u(y,t)\,\mathrm{d}\,y - u(x,t)\right)$$

3.7 Some TeXstudio tips

The reason we recommend TeXstudio is that it has many helpful tools to make LaTeX easier to navigate. If you navigate to LaTeX in the toolbar, you can select many of the common commands discussed above. Similarly `Math` will show you some of the most common mathematical functions.
For many commands TeXstudio provides an explanation and example – simply hover over the command or right-click it to open the package documentation.

[3] According to our local mathematician this describes that the rate of change of a plant density equals plant growth (dependent on plant and water density) minus plant death (constant rate B) and any (nonlocal) spatial interactions.

Chapter 4

Document Structure

4.1 Chapters, sections, subsections

LaTeX makes it easy to split your document into numbered parts. `Reports` have chapters, sections, subsections, subsubsections, and paragraph, while `articles` have all of these except chapters (hence: don't try to insert a chapter into an article - it will give you an error). A section heading is inserted with a command:

```
\begin{document}

\chapter{This is my first chapter}
\section{Here is a section}
\subsection{Here is a subsection}
\subsubsection{Here is a subsubsection}
\paragraph{Here's a paragraph with a title}

\end{document}
```

Per default LaTeX will number all sections and subsections - you can suppress the numbering by inserting an asterisk before the curly brackets:

```
\section*{Don't number this one}
```

Note that you can automatically create a table of contents, as well as lists of figures and tables:

```
\tableofcontents
\listoffigures
\listoftables
```

Titles and title pages can be created with a few simple commands. Note that the `\maketitle` command needs to go into the document body:

```
\title{title}
\author{names}
\date{text} OR \today

\maketitle
```

4.2 Cross-referencing

It is very easy to cross-reference anything (sections, tables, figures etc.) within your document. You can apply a `label` of your choosing to the section you want to reference. Note that in the case of figures and tables ("floats") this label MUST immediately follow the caption (if not it will refer to the section the float appears in, not the figure or table itself). You can then reference this label or the page it is on in your document:

```
\subsection{This subsection is relevant elsewhere}
\label{ImportantSubsection}

For more details see section \ref{ImportantSubsection} on page
\pageref{ImportantSubsection}.
```

If you use the package `hyperref`, each reference and label are further connected via hyperlink. The label is not printed, so you can call it whatever is most helpful for you. Instead of your label, LaTeX prints the number of the section or figure or table you attached the label to. You can further use the package `cleveref`, to make referencing a bit easier: the default command `\ref` will only return the number of your label (e.g. the number of the section, figure, table or page) but not the full reference (e.g. "section 3.2"). `cleveref` automatically figures out what type of object your label is attached to and prints it in front of the relevant number. All you need to do is load `cleveref` in your preamble and call references with `\cref` and`\cpageref` .

```
\subsection{This subsection is relevant elsewhere}
\label{ImportantSubsection}
```

```
For more details see \cref{ImportantSubsection} on
\cpageref{ImportantSubsection}.
```

You can move the object you have attached the label to around as long as you also move the label. To update all references, make sure to re-compile the document.

You can insert footnotes, which will be printed on the bottom of the page:

```
In linguistics, footnotes are not generally used.
\footnote{except to be snarky maybe}
```

4.3 Environments

LaTeX also makes use of `environments`, which are defined by a `begin` command and an `end` command:

```
\begin{environment}
Content of the evironment
\end{environment}
```

Environments include `document`, but also more specific things like lists (`enumerate`, `itemize`) and can be nested (and mixed):

```
\begin{enumerate}
\item Item 1
\begin{enumerate}
\item Subitem 1
\begin{itemize}
\item Subsubitem 1
\end{itemize}
\end{enumerate}
\item Item 2
\end{enumerate}
```

1. Item 1
    (a) Subitem 1
        - Subsubitem 1
2. Item 2

4.4 Large documents

If you have a very large document with several chapters (e.g. a dissertation), it might be useful to use the document class `report` rather than `article`. Unlike `articles`, `reports` contain several smaller files (e.g. one for each chapter). Only the main file contains the preamble, and the other files (saved in the same directory as the main file) are included using their file names with a command (the file extensions are optional):

```
\begin{document}

\tableofcontents

\chapter{Chapter 1}
\include{chapter1.tex}

\chapter{Chapter 2}
\include{chapter2.tex}

\chapter{Chapter 3}
\include{chapter3.tex}

\end{document}
```

Note that `\include` will force a page break before printing the content of the included files. If you want to avoid that you can use `\input`. However, one advantage of `\include` is that it allows you to select in the preamble which files should be included using `\includeonly{filename1, filename3}`.

In the context of large documents it might also be useful to know that you don't need to always fully compile your document (which might take considerable time). If you just want to check if your syntax is correct, you can use the package `syntonly` and the command `\syntaxonly` to get LaTeX to check syntax and commands without compiling a new PDF.

Chapter 5

References and Bibliographies

5.1 References in LaTeX

Referencing in LaTeX can seem a little daunting but once you've got the hang of it is really great. We promise. There are essentially three components to referencing in LaTeX, which we will consider in turn:

1. A .bib file which contains all your bibliography entries
2. A citation package (e.g. biblatex, cite, natbib)
3. A backend processor which matches your citation commands in the document with the bibliographical entries and does all kinds of magic (e.g. BibTeX, Biber)

Before we describe these in more detail, a terminological note. As noted above, there are two main backend options today, Biber and BibTeX. You might encounter people using the term "biblatex" in contrast to BibTeX (for instance – the people who made JabRef, see section 5.2.1), which you might find confusing because biblatex is a citation package and BibTeX is a backend processor. You are correct to find this confusing because it is. What people usually mean when they use biblatex in contrast to BibTeX is not just biblatex (the package) but *biblatex (the package) used in combination with Biber (the backend processor)*. Overall biblatex, in this sense, has largely replaced BibTeX (the backend processor running in concord with a package like natbib). The package is a lot more flexible, allows for more customisation and is better at handling non-Latin characters, which is why we would recommend using biblatex in combination with Biber.

5.2 Setting up a .bib file

The .bib file is a plain text file that contains all the bibliographical entries you will cite in your document. An entry looks, for example, like this (note that some of these fields are optional):

```
@article{ID,
author = {author},
title = {title},
journaltitle = {journaltitle},
date = {date},
OPTtranslator = {translator},
OPTannotator = {annotator},
OPTcommentator = {commentator},
OPTsubtitle = {subtitle},
OPTtitleaddon = {titleaddon},
OPTeditor = {editor},
OPTeditora = {editora},
OPTeditorb = {editorb},
OPTeditorc = {editorc},
OPTissuetitle = {issuetitle},
OPTissuesubtitle = {issuesubtitle},
OPTlanguage = {language},
OPToriglanguage = {origlanguage},
OPTseries = {series},
OPTvolume = {volume},
OPTnumber = {number},
OPTeid = {eid},
OPTissue = {issue},
OPTmonth = {month},
OPTpages = {pages},
OPTversion = {version},
OPTnote = {note},
OPTissn = {issn},
OPTaddendum = {addendum},
OPTpubstate = {pubstate},
OPTdoi = {doi},
OPTeprint = {eprint},
OPTeprintclass = {eprintclass},
OPTeprinttype = {eprinttype},
OPTurl = {url},
OPTurldate = {urldate},
}
```

```
@Article{Nguyen2016,
author = {Nguyen, Dong and
Do\v{g}ru\"oz, A. Seza and Ros\'e,
Carolyn P. and de Jong, Franciska},
title = {Computational Sociolinguistics:
 A Survey},
journal = {Computational Linguistics},
year = {2016},
volume = {42},
number = {3},
pages = {537--593},
doi = {10.1162/COLI\_a\_00258},
url = {https://doi.org/10.1162/COLI_a_00258},
}
```

The `ID` is your citekey, i.e. the string with which you will recall the bibliography

entry. This can be anything since it is not printed and only visible to you but it does make sense to establish some kind of system among your citekeys, perhaps of the format AuthorYYYY. There are many different types of entries depending on the format of your source such as articles, proceedings, chapters in edited volumes, books, online resources, technical manuals, etc. If you're using TeXstudio, you can click `Bibliography` in the toolbar above the editor and look at all entry types and the information they require.

5.2.1 Reference Manager

If you want, your .bib file can just be a plain text file but if you want your life to be both fun and easy we recommend using some kind of reference manager.

JabRef

JabRef is a free graphic interface to manage .bib entries. When you create a new .bib file (called a `library` within JabRef), you can select whether it is BibTEXor biblatex. As will be explained below, this is essentially a distinction depending on your backend. If you use BibTEX, select BibTEX, if you use biblatex, select biblatex. Next, give the file a name and save it into the same directory as your document – this way your document can find the .bib file very easily. Now you can start adding entries to your bibliography. Most online journals (and UoE DiscoverEd) offer an option to download the .bib-file for articles. You can download those and copy or import them into JabRef. For recent articles you can also use DOI or ArXiv IDs. If none of these are available (e.g. for older books) you can of course fill in all details by hand.

Mendeley

If you use Mendeley to manage journal articles you can use it to automatically generate BibTEXentries for all files you add. If you go to Tools > Options > BibTEX in Mendeley you can enable this feature and specify things like how many .bib files you want Mendeley to generate, where it should save them and whether it should automatically update them when you add new files.

5.3 Biblatex

There are essentially two main citation packages available today. One is natbib, which only works with BibTeX and the other is the more versatile biblatex, which is the one we will focus on here.

5.3.1 Citation commands

Biblatex has two citation commands:

```
\parencite[prenote][pagenumber]{ID}
\parencite[prenote][]{ID}
\parencite[pagenumber]{ID}
\parencite{ID}
\textcite[see][pagenumber]{ID}
```

\parencite gives you a citation in parenthesis (in this style is this Surname, Year). Note that the optional parameters in the square brackets are optional and can be left out, but if you do use them be aware that optional parameter before the citation key will be interpreted as the page number (no need to spell out the <p>, just insert a number). If you only want a prenote, just leave the second set of brackets empty:

```
Many people have said that \LaTeX{} is great
\parencite[see e.g.][][Markl2019, Papineau2019].
```

Many people have said that LaTeX is great (see e.g. Markl 2019, Papineau 2019).

```
\LaTeX{} has been argued to be highly useful for typesetting linguistics
 \parencite{Markl2019}.
```

LaTeX has been argued to be highly useful for typesetting linguistics (Markl 2019).

```
However, some people have said that their friends have ``bullied them
into using it'' \parencite[23]{Papineau2019}.
```

However, some people have said that their friends have "bullied them into using it" (Papineau 2019, p. 23).

`\textcite` gives you the name of the author in your main text, followed by year in parentheses:

Markl (2019) argues that LaTeX is great and a wonderful way to feel productive while trying to fix error messages.
Papineau (2019, p. 44) claims that he is "sick of being used to illustrate citation styles".

5.4 Backend

The backend processor can map your citation command to the correct entry in the .bib file. To use the full functionality of biblatex, use Biber.

5.5 Putting it all together

In order to use your bibliography you need to define your .bib file, the package you want to use, as well as the bibliography style (of which there are many - including one following the unified stylesheet for linguistics `biblatex-langsci-unified`) and the backend in your preamble. Luckily, there are specific styles for linguistics (which are included in TeXLive).

```
\usepackage[bibstyle=style, citestyle=style, backend=backend]{citation-package}
\addbibresource{bibliographic resource}
```

```
\usepackage[citestyle=langsci-authoryear-comp,
bibstyle=biblatex-langsci-unified, backend=biber]{biblatex}
\addbibresource{bib.bib}
```

Then you can cite any entries in your .bib file in your document. To print your bibliography in your desired citation style with the command `\printbibliography`.

If you want to further modify the bibliography style, for example because you don't want your bibliography to print certain fields of your entry, you can tell LaTeX in the optional parameter of the citation package (or create your own style - that's beyond the scope of this handbook but the package documentation explains how):

```
\usepackage[citestyle=langsci-authoryear-comp,
bibstyle=biblatex-langsci-unified, backend=biber, doi = false]{biblatex}
```

The command `doi = false` prevents LaTeX from printing the DOI in your bibliography - you can use these commands to block any field from being printed.

Chapter 6

Packages

6.1 Floats

Floats are environments that LaTeX does not break up across pages. They are not part of the normal text and are always numbered and can be referred to at any point in the text using the `label` and `ref` commands. Figures and tables are inserted as floats (tabulars don't have to be inserted as floats but if they are not you cannot refer back to them and you cannot move them to a different place in your document). While you can use the built-in table and figure environments, there are specific packages that allow for more modification.
LaTeX automatically places floats wherever there is space - if there is not enough space on the current page it will push it to a new page. The automatic placement can be overridden by specific placement commands:

Spec	**Placement permission**
h	right here
t	at the top of a page
b	at the bottom of a page
p	on a page with other floats
!	ignore internal specifications such as maximal number of floats on a page

Table 6.1: Placement specifications for floats.

You can also force LaTeX tp print all floats you have defined so far via the command `\clearpage`.

6.1.1 Tables: tabular, longtable

LaTeX is not a spreadsheet. Constructing a big table from scratch in LaTeX can therefore be a bit annoying, but there are many ways to import tables from other programmes such as R or Excel. We will first cover the basics of how to create tables from scratch before showing you some of the shortcuts.

The actual rows and columns of a table are defined in the `tabular` environment. If you put this `tabular` into a `table` environment (a floating environment), you will be able to refer back to it, add captions, etc.

```
\begin{tabular}[pos]{table spec}
\end{tabular}
```

`table spec` defines the number and alignment of columns. `\hline` gives you a horizontal line (don't forget to insert a line break before the horizontal line).

```
\begin{tabular}[c]{|l l|} \hline
command & output  \\ \hline
l & left-aligned  \\ \hline
r & right-aligned  \\ \hline
c & centred \\ \hline
\end{tabular}
```

command	output
l	left-aligned
r	right-aligned
c	centred

```
\begin{tabular}[c]{l l}
command & output  \\
l & left-aligned  \\
r & right-aligned  \\
c & centred \\
\end{tabular}
```

command	output
l	left-aligned
r	right-aligned
c	centred

You can (and should) add a caption to your table using the command `\caption`. Note that these need to be outwith the `tabular` environment but inside the `table` environment (all captions need to be inside a "float"). Every table is automatically labelled but you can use `hyperref` to create a `label` which you can refer back to in the text (with a hyperlink). `\centering` makes sure your table is in the centre of the page:

```
\begin{table}[h!]
\centering
\begin{tabular}[c]{|l l|} \hline
command & output  \\ \hline
l & left-aligned  \\ \hline
```

```
r & right-aligned  \\ \hline
c & centred \\ \hline
\end{tabular}
\caption{This is a tabular showing you how to tabular (table spec).}
\label{tabular}
\end{table}
```

command	output
l	left-aligned
r	right-aligned
c	centred

Table 6.2: This is a tabular showing you how to tabular (table spec).

You can also have multiple columns under the same header (and the caption can also be above your tabular):

```
\begin{table}[ht!]
\centering
\caption{This is a complicated table describing complicated things.}
\begin{tabular}{|l l|} \hline
\multicolumn{2}{|c|}{\textbf{Complicated Table}} \\ \hline
Column 1 & Column 2 \\ \hline
Entry 1 & Entry 2 \\ \hline
\end{tabular}
\end{table}
```

Table 6.3: This is a complicated table describing complicated things.

Complicated Table	
Column 1	Column 2
Entry 1	Entry 1

If you have a very long table that extends over several pages you can use the package `longtable`. If you want to import a table from Excel you can use a Excel macro like Excel2LaTeX. You can produce more elegant tables using the package `booktabs`.

Instead of using the commands `\hline` to insert horizontal lines you simply define the top horizontal line and its width ("toprule"), the middle horizontal lines and their width ("midrule") and the bottom horizontal line ("bottom rule"):

```
\begin{table}[h!]
\begin{tabular}{l l}
\toprule
command & output \\
\midrule
l & left-aligned \\
r & right-aligned \\
c & centred \\
\bottomrule
\end{tabular}
\caption{This is more elegant version of the tabular above using \texttt{booktabs}.
\end{table}
```

command	output
l	left-aligned
r	right-aligned
c	centred

Table 6.4: This is more elegant version of the tabular above using `booktabs`.

6.1.2 Inserting images into figures: graphicx and TikZ

Images are also inserted into floating bodies in LaTeX. One very useful package in this context is `graphicx`. Usually images are inserted into `figures`. To insert a graphic use the command `\includegraphics[width OR height]{filename}`. Note that any labels you want to attach to the figure must immediately follow the caption. Make sure that the files are in the same folder as your TeX file (or else provide the full path).

```
\begin{figure}
\includegraphics[width=\textwidth]{hyrax.jpg}
\caption{A hyrax at the San Diego Zoo.}\label{hyrax}
\end{figure}
```

Which looks like this[1]:

Figure 6.1: A hyrax at the San Diego Zoo.

[1] `https://animals.sandiegozoo.org/animals/rock-hyrax`

You can also insert several graphics into one figure via `subfloats` using the package `subfig`.

```
\begin{figure}[h!]
\subfloat[Closely related to the hyrax.]
{\includegraphics[width=0.5\textwidth]{Elephant.jpg}}
\hspace{1cm}
\subfloat[Not closely related to the hyrax.]
{\includegraphics[width=0.5\textwidth]{guineapig.jpg}}
\end{figure}
```

(a) Closely related to the hyrax.

(b) Not closely related to the hyrax.

Importing graphics from R

If you're importing graphics from R created with `ggplot`, you can use the command `ggsave` in R to save a file and then just add it as outlined above. Alternatively you can also use the R package `tikzDevice` and the LaTeX package `TikZ` to incorporate `ggplot` graphics in your document. The advantage of this latter method is that it will automatically update any changes you make to your `ggplot` graphic in LaTeX:

1. Load the package `tikzDevice` in R and the package `TikZ` in LaTeX
2. Before defining your ggplot, insert `tikz(file = yourfile.tex, width = yourwidth, height = yourheight)` – this turns the following ggplot into a .tex file
3. After the ggplot code, close the environment using the command `dev.off()`
4. Input the .tex file into a figure in your LaTeX document:

```
\begin{figure}
\centering
\input{ggplot.tex}
\caption{Insert caption here \label{fig:ggplot}}
\end{figure}
```

6.2 Miscellaneous packages

Documentation for all packages can be found at CTAN. Here are some of the most useful[2] packages.

- `hyperref`: automatically includes internal and external hyperlinks in references (e.g. cross-referencing, urls, citations and bibliography entries)
- `cleveref`: automatically identifies which type of object you are referring to (section, figure, table, page etc.) and adds that information in your text
- `todonotes`: allows you to add annotations to the output file as you're working on it and generate a "to-do-list"
- `setspace`: allows you to customise margins and line spacing
- `fancyhdr`: Let's you customise headers and footers
- `babel`: language support (works with all engines and compilers)
- polyglossia: language support alternative to `babel` (works only with XeLaTeX and LuaLaTeX)
- `csquotes`: context-sensitive quotation management (adjusts to the language set in `babel`)
- fontspec: main package used in conjunction with XeLaTeX to specify fonts
- `multicol`: allows you to create multicolumn environments (also within tabulars)
- `TikZ`: in addition to importing graphics from R, TiKZ also lets you draw pictures with vectors, circles, arrows and other shapes (mostly used by mathematicians)
- `amsmath`: standard package for mathematics
- `graphicx`: enhanced support for inserting graphics (e.g. optional parameters in `\includegraphics`)

[2] well...

- `longtable`: great for long tables extending over several pages
- `booktabs`: produces nice-looking tabulars in the tabular environment
- `xcolor`: lets you add colour to your document - you can define colours in the preamble
- `verbatim`: typesets your content verbatim, i.e. without implementing any commands (this is the package used in this document to display the code).
- `soul`: allows you to ~~strike out~~ text
- `coffee`: Did you know that you can save time and money if you print coffee stains on your documents rather than add them manually?

6.3 Linguistic packages

You can browse some of the packages commonly used in linguistics on `https://ctan.org/topic/linguistic`.

6.3.1 Phonetics and Phonology

tipa

In addition to using unicode input to typeset IPA symbols in XeLaTeX, there is also a specific package to typeset IPA symbols, called `tipa`. By declaring the `\usepackagetipa` package in the preamble, you will be able to use these symbols by way of LaTeX commands. There are a few ways of doing this, as outlined below.

Individual Commands

The first way of doing this is to just make individual commands, which is nice if you only need one or two symbols in your paper. For example, if you need the glottal stop, you can sImply use the shortcut commad defined by tipa, found in the TIPA manual (a document you will use a lot!). `\textglotstop` produces ʔ. You can also use this package to add diacritics to normal characters, as well as other TIPA characters. For example, you can use `\textsubbridge{t}` to produce t̪ with a dental diacritic. Or, you could use `\textsubbridge{textglotstop}` to produce ʔ̪ (though

of course this symbol makes little sense). If you are going to use this package this way, make sure you leave spaces between commands. For example,

```
\textglotstopa
```

will not produce ʔa, but will return an error. You should make sure to type it as

```
\textglotstop a
```

to produce the correct form.

The IPA environment

Another option for using TIPA is to use the tipa environment, and to use the shortcuts thusly defined in the tipa manual. An example is reproduced here:

```
\begin{IPA}
P
\end{IPA}
```

which would produce ʔ . This is particularly useful for larger chunks of IPA encoding. Note that the shortcuts are different from those used in the simple command encoding, and to be sure you're using the correct one.

The textipa command

The final option for TIPA is to use the

```
\textipa{...}
```

command. In this environment you can insert the same shortcuts you would use in the IPA environment. For example,`\textipa{P}` produces: ʔ. This is particularly useful for longer words that need to be rendered in IPA. You can also do this with individual commands, but the results may be unwieldly:

```
d\textsci .\textprimstress bi.n-\textschwa
```

to produce [dɪ.ˈbi.n-ə].

OT Tableau

The ot-tableau package is a LaTeX package used for the creation of Optimality Theory tables. You can include this package by inserting the `\usepackage{ot-tableau}` command in your preamble. This is done with the `{tableau}` environment, then defined as you would define table columns (see 6.1.1). An example is provided here:

```
\begin{tableau}{c:c|c}
\inp{\ips{stap}} \const{*Complex} \const{Anchor-IO} \const{Contiguity-IO}
\cand{stap} \vio{*!} \vio{} \vio{}
\cand[\Optimal]{sap} \vio{} \vio{} \vio{*}
\cand{tap} \vio{} \vio{*!} \vio{}
\end{tableau}
```

This example will provide the OT table:

/stap/	*Complex	Anchor-IO	Contiguity-IO
a. stap	$*!$		
☞ b. sap			$*$
c. tap		$*!$	

As you can see, the constraints are labeled in the `\const{label}` arguments, and the `\cand` commands are used to provide the various options to be tested under the OT logic. If you'd like to label the most optimal option, you can use the `[\Optimal]` argument after the `\cand` command.

PhonRule

Phonrule is a LaTeX package (loaded by commanding `\usepackage{phonrule}` in the preamble.) that allows for the creation of phonological rules. This is done by using the `\phon{/x/}{[y]}` command, where 'x' is the phoneme undergoing the change and 'y' is the form that is realised. For example,

```
\phon{/t/}{[\textglotstop]}
```

will produce the phonological rule /t/ → [ʔ].
If you need to add context, you can use `\phonc{/x/}{[y]}{[environment]}` to add a third argument. You could use `\phonc{/t/}{[\textglotstop]}{[-stress]}`, for example, to produce the rule /t/ → [ʔ] / [-stress]. If you need to add _ signs for

the context, you can use the commands `\phonl, \phonr` and `\phonb` to add one on the left (l), on the right (r) and on both sides (b). The commands

```
\phonl{/t/}{[\textglotstop]}{i}
\phonr{/t/}{[\textglotstop]}{i}
\phonb{/t/}{[\textglotstop]}{i}{i}
```

will produce the rules

/t/ → [ʔ] / V_
/t/ → [ʔ] / _V
/t/ → [ʔ] / V_V

If you need multiple possible environments for the change, you can use the `\oneof` command to specify this. An example is provided here:

```
\phonc{t}{\textglotstop}{
\oneof{
\phold i \\
\phold u}
}
```

This example will produce:

$$\text{t} \rightarrow \text{ʔ} \; / \; \left\{ \begin{array}{l} \text{\_i} \\ \text{\_u} \end{array} \right.$$

The `\phold` command here places the _ placeholder marker in the environment. If you need to specify features in the environment part of the phonological rule, this can be done with the `\phonfeat` command, exemplified here:

```
\phonc{t}{\textglotstop}{\phold
\phonfeat[l]{-
consonantal \\
+high \\
+front}
}
```

Which will produce the rule:

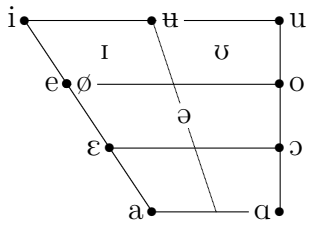
$$\mathrm{t} \rightarrow \mathrm{ʔ} \; / \; \_ \begin{bmatrix} \text{- consonantal} \\ \text{+high} \\ \text{+front} \end{bmatrix}$$

Vowel

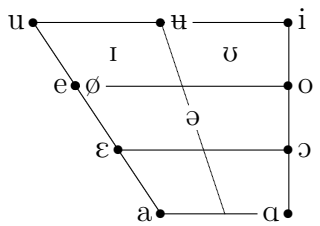
Vowel is a LaTeX package (loaded by commanding `\usepackage{vowel}` in the preamble) that allows you to insert vowel trapeziums that place the vowels in a particular language on said trapezium. This is done by defining the vowel environment and telling LaTeX which vowels you'd like to appear, and in which positions. An example is given here:

```
\begin{vowel}
\putcvowel[l]{i}{1}
\putcvowel[l]{e}{2}
\putcvowel[r]{\o}{2}
\putcvowel[l]{\textepsilon}{3}
\putcvowel[l]{a}{4}
\putcvowel[l]{\textscripta}{5}
\putcvowel[r]{\textopeno}{6}
\putcvowel[r]{o}{7}
\putcvowel[r]{u}{8}
\putcvowel[r]{\textbaru}{9}
\putcvowel{\textschwa}{11}
\putcvowel{\textsci}{13}
\putcvowel{\textupsilon}{14}
\end{vowel}\\
```

This text will produce



The [l] or [r] arguments in each command tell LaTeX which side of the node to put the vowel on, and the number indicates the location on the vowel trapezium. Generally you don't want to change which vowel appears where, else you end up with something looking like:



Where the high vowels /i/ and /u/ are in the wrong backness categories. Also worth mentioning here is that the `vowel` package relies on `tipa`, so make sure if you're going to use it that you have both declared.

6.3.2 Syntax

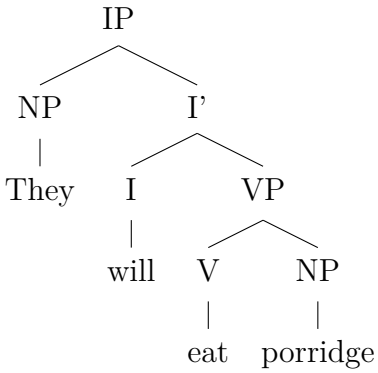
qtree

The qtree package is used for syntactic trees in LaTeX, and is implemented by declaring the `\usepackage{qtree}` command in the preamble of your document. It's a powerful package, but we will cover only the basics here, which should be enough for your undergraduate degree.

The command is initiated with the `\Tree` command, followed by a space before you begin to bracket out your tree. When bracketing, remember that anything appearing within the same set of brackets will be connected. At the lowest level, e.g. [.NP porridge], An example is provided here:

```
\Tree [.IP [.NP They ] [.I' [.I will ] [.VP [.V eat ] [.NP porridge ] ] ] ]
```

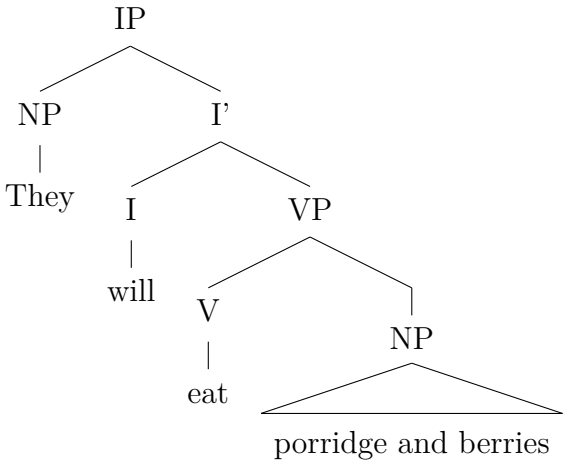
This will produce the following tree:



If you need to use a multi-orthographic node, you can use the `\qroof{text}` command, followed by the label for the node. Note that this is the opposite of single-orthographic nodes, and producing these commands in the wrong order will produce an error. An example is provided here.

```
\Tree [.IP [.NP They ] [.I' [.I will ] [.VP [.V eat ] [\qroof{porridge and berries}
```

This will produce the following tree:



As with other lingusitics packages, you can use things like tipa inside of the tree, just as you would normally in regular text. You can also use commands such as `\st` (a command loaded through `\usepackage{soul}`) to create strikethroughs for things such as movement.

6.3.3 Morphology

gb4e

gb4e is a package in LaTeX that allows for the insertion of linguistic examples, particularly for morphological glossing. You insert the `\usepackage{gb4e}` command in your preamble, but the gb4e package is a bit finnicky and demands that it be the last package declared in the preamble. If you don't do this, TeXstudio will return an error, so just be careful.

The primary use of gb4e is for creating interlinear glosses, which occur within the exe environment. These glosses have three parts: the data to be glossed, the morphology gloss, and the translation. An example of this environment, with labelled parts, is provided here:

```
\begin{exe}
\ex
\gll {\textepsilon z} we d\textsci .\textprimstress bi.n-\textschwa m  \\ <- data t
1SG.NOM 3SG.FEM.OBL see-1SG \\ <- morphological gloss
\trans `I see her' <- translation
\end{exe}
```

This will provide the following gloss:

(1) ɛz we dɪ.ˈbi.n-əm
1SG.NOM 3SG.FEM.OBL see-1SG
'I see her'

The use of the exe environment will also automatically label the examples for you, so you don't need to worry about that. If, for some reason, you need to change the example number, however, you can do so by replacing the `\ex` command inside of the exe environment with `\exi{label}` and change the label to what you'd like. An example is provided here:

```
\begin{exe}
\exi{(3)}
\gll {\textepsilon z} we d\textsci .\textprimstress bi.n-\textschwa m  \\
```

```
1SG.NOM 3SG.FEM.OBL see-1SG \\
\trans `I see her'
\end{exe}
```

This code will provide the following gloss, not automatically labelled:

(3) ɛz we dɪ.ˈbi.n-əm
1SG.NOM 3SG.FEM.OBL see-1SG
'I see her'

You can also input subexamples in more general examples, by using the `\begin{xlist}` environment. An example (pun absolutely intended) is introduced here:

```
\begin{exe}
\ex
\begin{xlist}
\ex I am grammatical
\ex *Grammatical not I am
\end{xlist}
\end{exe}
```

This set of code will produce the following numbered and sub-labelled example:

(2) a. I am grammatical
b. *Grammatical not I am

Chapter 7

Troubleshooting

7.1 Quick fixes

Sometimes things don't go as planned - here are some quick fixes you can try when your file does not compile properly:

- compile again (F5)
- if you have a bibliography file: re-run biber/BibTeX (F8)
- clean auxiliary files (Tools > Clean Auxiliary Files ...) and compile again
- check if all brackets are closed (especially nested brackets!)
- check for spelling errors in commands
- make sure that there are no incomplete commands (e.g. LaTeX really dislikes an empty `\item`)

7.2 Bibliography

Bibliography files are a common source of errors.

- If the "Messages" tab indicates a specific entry - check that all brackets in this entry are closed and commas are places correctly
- Similarly make sure that all special accents and characters are properly encoded
- A common source of error in bibliography entries are hyphens in the `pagenumber` field - make sure those are hyphens (`--`) rather than minuses (`-`).

Chapter 8

Further reading

8.1 For an introduction to LaTeX:

- https://en.wikibooks.org/wiki/LaTeX
- http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf
- https://tobi.oetiker.ch/lshort/lshort.pdf

8.2 For an introduction to LaTeX for linguistics specifically:

- https://en.wikibooks.org/wiki/LaTeX/Linguistics
- https://www.linguisticsociety.org/content/using-latex-linguistics
- https://adamliter.org/content/LaTeX/latex-workshop-for-linguists.pdf

8.3 Troubleshooting:

- https://tex.stackexchange.com/

8.4 Documentation:

- https://ctan.org/topic/linguistic
- https://ctan.org/