A crash course in \LaTeX{} (for linguistics)

COPiL editorial team 2021-3

Ema Banerjee, Alex Cairncross
Nina Haket, Sana Kidwai

The University of Cambridge

20 Jan 2023
Want to follow along?

Check email for slides or visit website
Overview

Why \LaTeX?  
Basic document structure  
Referencing  
Break (20 min)  
Floats  
Trees  
Special characters (IPA / variables)  
Glossing  
\COPiL template
Why \LaTeX?
Why \LaTeX?: Linguistics reasons

Trees

(1) \[ \begin{array}{c} X \\ Y \\ Z \end{array} \]

(2) \[ \begin{array}{c} X \\ Y \\ Z \end{array} \]

Numbered and or glossed examples

(3) \[ \begin{array}{l} \text{L'=}ha \\ \text{CL=}has \\ \text{Gianni} \end{array} \]

‘Gianni ate it.’

[Italian]

(Cross-)referencing (e.g. 1-3)

Language other than English/ special characters (e.g. è, é, ě, Ė, ε, 穴, ə, з ... )
Why \LaTeX?: Your own sanity

Automate repetitive tasks
- Bibliographies
- List of figures/tables
- Custom commands

Reduce human error
- Referencing
  - First vs ‘et al.’
  - Missing entries
  - Alphabetising
  - Formatting

Formatting and content stored separately
- No more checking for double spaces / unintended indents
Why \LaTeX?: Other considerations

Ease of customisation

Stack Overflow and Overleaf (docs)

Your poor computer

Stability

Forgetting about formatting
Beyond articles or your thesis

Handouts
- Supervisor meetings
- (Syntax) talks

Posters
- Conference
- Participant recruitment

Presentation slides (like these)

Job applications
- CV
- Cover letter
Caveat

\LaTeX{} takes an initial investment

Word/Google Docs has a time and place
- You have an abstract due in an hour and you don’t have a template
- Extra features are irrelevant (e.g. COPiL meeting notes)
- Some journals require word documents (double check!)
- Your collaborator/supervisor may prefer editable word documents rather than pdfs
Basic Document Structure
Basic Document Structure

Setting up a document

Preamble and packages

Basic formatting

Lists

Sections

Troubleshooting
Setting up a document

Create an Overleaf account (or just log in via Google) if you haven’t already.

Logging in will take you to the ‘Projects’ page.

Click New Project → Blank Project.

Enter a title and create the project.
Setting up a document

Overleaf does a lot of the basic work for you.
Preamble

**Preamble:** the ‘setup’ section of a document.

Includes everything before `\begin{document}` (where the actual document contents begins).

Functions:

- Defines document class (article, book, etc).
- Configures the document - languages, page setup, etc.
- Loads packages.
- And lots more! You can create your own commands, define specific colours, make a new environment type, and so on.
Packages

**Packages:** external bodies of code that provide specialist capabilities or extend \LaTeX{}’s built-in features.

Packages are distributed through CTAN, which currently has 6365 available.

Imported using the command
\begin{verbatim}
\usepackage[<options>]{<packagename>}
\end{verbatim}

Some examples:
- \begin{verbatim}
\usepackage{graphicx}
\end{verbatim} - for using image files.
- \begin{verbatim}
\usepackage[english,italian]{babel}
\end{verbatim} - for using (multiple) languages/alphabets.
- \begin{verbatim}
\usepackage[table,dvipsnames]{xcolor}
\end{verbatim} - for using colours, including by name and in tables.
Basic formatting: formatting text

Bold, italics, underlining, superscript, subscript, small caps.

Commands:
- \textbf{}
- \textit{} or \textsl{}
- \underline{}
- \textsuperscript{}
- \textsubscript{}
- \textsc{}``
Basic formatting: formatting text

1 Text can easily be $\textbf{bolded}$, $\textit{italicised}$, and $\underline{underlined}$.  
2 You can also add superscript text $\textsuperscript{like this}$, subscript text $\textsubscript{like this}$, or text in $\textsc{small caps}$.

Text can easily be **bolded**, *italicised*, and *underlined*. You can also add superscript text **like this**, subscript text **like this**, or text in **SMALL CAPS**.
Basic formatting: font sizes

There are ten default font sizes: \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge and \Huge (note that capitalisation matters!).

These sizes are relative to the base fontsize (which can be defined in the preamble) - you can also use absolute sizes if you prefer.

They affect whatever text follows them, until another command or a different environment is defined.

1 \Huge Very big text, \Large large text, \small small text, \tiny tiny text.

2 small text, \tiny tiny text.
Basic formatting: paragraphs and spaces

Paragraphing and other spacing can be done in a few ways.

Line breaks can be inserted by leaving a blank line, typing `\`, or using `\newline`.

Vertical spaces can be added using `\vspace{<length>}`, or the commands `\smallskip`, `\medskip`, and `\bigskip`.

Horizontal spaces can be added using `\hspace{<length>}`, or `\hfill` to move everything to the other side of the line.
Basic formatting: paragraphs and spaces

1 This is a paragraph. \\
2 This is a line with a comment. \hfill This is the comment.
3
4 \vspace{2cm} This is a line 2cm below the last one.

This is a paragraph.
This is is a line with a comment. This is the comment.

This is a line 2cm below the last one.
Lists

Example of an *environment*, which applies specific typesetting effects to just that part of the document. Environments are contained in `\begin{<name>}` and `\end{<name>}` tags.

Unordered (bullet point) lists use `itemize`, ordered (numbered) lists use `enumerate`.

Individual items use `\item`.

Can be nested to create sublists.

Environments must be closed in relative order; most recently opened first.
Lists

1 \begin{itemize} \item An item in an unordered list. 
2 \item Another item, introducing a numbered sub-list. 
3 \end{itemize} 
4 \begin{enumerate} \item The first item in this sub-list. 
5 \item A second item. 
6 \end{enumerate} 
7 \end{itemize}

- An item in an unordered list.

- Another item, introducing a numbered sub-list.
  1. The first item in this sub-list.
  2. A second item.
Lists: examples

\texttt{enumerate} restarts the numbering every time...not great for linguists.

Solution: packages! There are several specific packages for formatting examples (with continued numbering, sub-parts, etc.) including \texttt{linguex} and \texttt{expex}.

Packages that try and do the same thing don’t work nicely together - you should stick to one or another.

\begin{verbatim}
1 \texttt{pex} \a This is the first sentence.
2 \a This is another sentence in the same example.
3 \texttt{xe}
\end{verbatim}

(4) \texttt{a.} This is the first sentence.
\texttt{b.} This is another sentence in the same example.
Sections

Sections (and subsections, subsubsections etc) divide documents.

There are multiple levels of depth, depending on the document class.

Sections do a lot of things automatically:
- Numbering (and cross-referencing)
- Formatting headers
- Creating a contents page (using `\tableofcontents`)

`\section{Title}` creates a section, `\subsection{Title}` and `\subsubsection{Title}` create nested (sub)subsections.

Unnumbered sections can be created by adding an asterisk at the end of the command.
Sections

1 \section{Section One}
2 \subsection{A subsection}
3 This is just some generic text in the subsection, enough to fill a line or so.
4 \subsection*{Another, unnumbered subsection}
5 Some more text in another subsection.

1 Section One
1.1 A subsection
This is just some generic text in the subsection, enough to fill a line or so.

Another, unnumbered subsection
Some more text in another subsection.
Troubleshooting

If there’s an issue with the code, \LaTeX{} can usually figure out what it is (or at least give you the line where it occurs).

Issues can either be warnings (shown in orange, usually auto-corrected, can generally be ignored) or actual errors (shown in red, can prevent compiling, should be fixed).

Other resources: Overleaf documentation, StackExchange, Google.
Basic document structure: your turn

Create some sections (and subsections if you want!) in your document, for example the sections in today’s presentation (this section, referencing, floats, trees, special characters, glossing).

- \section{...}

Write some text in your introduction and format it.

- e.g. \textbf{...}, \textit{...}, etc.

Summarise the material in this section using a list.

- \begin{itemize}...
- or \begin{enumerate}...
  (don't forget to end the environment).

Import a package; you can pick one from CTAN, or add the graphicx package.

- \usepackage[<options>]{<packagename>}

Referencing
Referencing

Cross-referencing

External links

Bibliography
Basic cross-referencing

Referencing sections, examples, tables, figures etc. in the document.

Commands:
- `\label{<label>}`
- `\ref{<label>}`

```
1 \section{Section 1} \label{sec1}
2
3 A reference to section \ref{sec1}.
```

1 Section 1

A reference to section 1.
hyperref package

The hyperref package creates hyperlinks.

Preamble:
- `\usepackage{hyperref}`
- `\hypersetup{`
  - `colorlinks=true, (links will be coloured, default is red)`
  - `linkcolor=blue, (colour for internal links)`
  - `urlcolor=blue, (colour for URLs)`
  - `citecolor=blue, (colour for citations)`
  - `allcolors=blue, (colour for all types of links)`
  }

Cross-referencing with \texttt{hyperref}

Commands:
- \texttt{\ref{<label>}}
- \texttt{\autoref{<label>}}

\begin{verbatim}
\section{Section 1} \label{sec1}
\end{verbatim}

1 A reference to \texttt{\ref{sec1}} vs. \texttt{\autoref{sec1}}.

\section{Section 1}

A reference to 1 vs. \texttt{section 1}.
External links with \texttt{hyperref}

URLs:
- \url{<url>}
- \href{<url>}{<text>}

1 \url{https://www.mml11.cam.ac.uk/cambridge-occasional-papers-linguistics}
2
3
4 \href{https://www.mml11.cam.ac.uk/cambridge-occasional-papers-linguistics}{The COPiL website}

\url{https://www.mml11.cam.ac.uk/cambridge-occasional-papers-linguistics}
The COPiL website
External links with \texttt{hyperref}

Email addresses:

- \href{mailto:<email address>}{<text>}

1 Click \href{mailto:copil@mml1.cam.ac.uk}{here} to email the COPiL team!

Click \texttt{here} to email the COPiL team!
Bibliography

\LaTeX{} can automatically format citations, both in-text and in the bibliography.

The bibliography information is stored in a .\texttt{bib} file.

The entries in your .\texttt{bib} file can be referenced in the document.

Opening a .\texttt{bib} file:
- Click \textit{New File} (left-most icon under Menu).
- Enter a file name.
- Change the file extension to .\texttt{bib}. 
.bib file

Each bibliography entry is stored as a .bib entry in the following format:

@article{key, (@entry type, key = label)
    author = {},
    title = {{}}, (double brackets to protect caps, also for booktitles)
    journal = {},
    year = {},
    volume = {},
    number = {},
    pages = {},
    note = {},
}

A useful masterlist: https://www.bibtex.com/e/entry-types/
Bibliography

Preamble:
- \usepackage{natbib}
- \bibliographystyle{apastyle} (to set the referencing style)

Document: \bibliography{<bib file name>.bib}
(to insert the bibliography, usually near the end of the document)

As a default, only the entries you cite in-text will be printed in the bibliography.
In-text citations

Commands:
- $\texttt{\cite{<key>}}$
- $\texttt{\citep{<key>}}$

You can find the rest of the commands in the \texttt{natbib} package documentation.

\begin{verbatim}
1 \cite{Chomsky1980}
2 \citep{Chomsky1980}
\end{verbatim}

Chomsky (1980)
(Chomsky, 1980)

Note: If you are using the \texttt{hyperref} package, all in-text citations will automatically link to the full reference in the bibliography.
In-text citations

To add information to an in-text citation:
\citep[<before>][<after>]{<key>}

1 \citep[see][p.1]{Chomsky1980}
2 \citep[p.1]{Chomsky1980}
3 \citep[see][]{}{Chomsky1980}

(see Chomsky, 1980, p.1)
(Chomsky, 1980, p.1)
(see Chomsky, 1980)
Referencing: your turn

1 First Section

In this section, we’re going to practise referencing (1).

1. Submit to COPiL!

Chomsky (1980, p.1-5) said some stuff (see also Chomsky, 1981, 1991). All of Chomsky’s ideas are discussed in section 2.

2 Second Section

Something about linguistics.

References


Break
Floats
Floats

Unit that should stay together, but maybe not right here
- Tables
- Figures
- Custom

Separation of formatting and content

General pattern: top, bottom, next top


Import an image

Package: graphicx

Command: `\includegraphics{<path to file>}`

Option: specify width/height in your favorite unit (cm, in, % of textwidth . . . )

```
1 \includegraphics[width=.35\textwidth]{Images/Procrastination.jpg}
```

![Image with text: Something I've been putting off for weeks that wouldn't take longer than fifteen minutes to complete.](Images/Procrastination.jpg)
Images as figures

Stand-alone image is not nicely formatted not useful (no caption / no crossref)

Solution: Figure environment

\begin{figure}[<position>]
  \centering
  \includegraphics[...]{...}
  \caption{<caption>}
  \label{<label>}
\end{figure}

Basic positions: \texttt{t[op]}, \texttt{b[ottom]}, \texttt{h[ere]}

Caption must proceed label!

Tip: indent for future you
Example figure

\begin{figure}
  \centering
  \includegraphics[width=.5\textwidth]{Images/...}
  \caption{Me on a daily basis}
  \label{Me}
\end{figure}
Figure 1: Me on a daily basis
Tabular environment

\texttt{tabular} \neq \texttt{table}

- \texttt{tabular} environments create collections of cells ← ‘\texttt{includegraphics}’
- Tables (containing \texttt{tabulars}) float ← ‘\texttt{figure}’

Basic tabular case
\begin{tabular}{<columns>}
  <contents> & <contents> \\
  <contents> & <contents> \\
\end{tabular}

column alignments: \texttt{c[enter]}, \texttt{r[ight]}, \texttt{l[eft]}

lines (for top/bottom/midrule: \texttt{\usepackage\{booktabs\}})
  - vertical: \texttt{|} in column specifications
  - horizontal: \texttt{\toprule, \bottomrule, \midrule} in between rows
Tabular with rule lines

\begin{tabular}{r|cc}
\toprule
& Control & Experimental \\ 
\midrule
Condition A & 10 & 15 \\ 
Condition B & 17 & 16 \\ 
\bottomrule
\end{tabular}
Tabular and merging

Package: multicol

Merge horizontally: \multicolumn{<# cols>}{<align>}{<content>}

Merge vertically: \usepackage{multirow} ← preamble
\multirow{<# rows>}{<width>}{<content>} ← ‘*’ = inherit width

Merge both?: you can nest multi row in multi column … but not often required

Wrap table environment like figure to make float
Tabular multicols and cmidrule example

\begin{table}
\centering
\begin{tabular}{ r cc }
\toprule
& \multicolumn{2}{c}{Control} \\
\cmidrule{2-3}
 & Mean & SD \\
\midrule
Condition A & 10 & 3 \\
Condition B & 17 & 4 \\
\bottomrule
\end{tabular}
\caption{Some data}
\end{table}
Tabular multicols and cmidrule example again

<table>
<thead>
<tr>
<th>Control</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition A</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Condition B</td>
<td>17</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Some data
Table Generators

Figure 2: www.tablesgenerator.com
OT tableaux

Define \textbf{hand}: \texttt{\usepackage{pifont}}
\texttt{\newcommand{\hand}{\ding{43}}}

<table>
<thead>
<tr>
<th>/stap/</th>
<th>*COMPLEX</th>
<th>ANCH.-IO</th>
<th>CONT.-IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. stap</td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. sap</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>c. tap</td>
<td></td>
<td></td>
<td>*!</td>
</tr>
</tbody>
</table>

Figure 3: Example of an OT tableau

Basic tabular is a bit tedious . . .

Instead use a custom package like \texttt{ot-tableau} (used to make figure above)
OT tableaux again

1 \ShadingOn
2 \begin{tableau}{c:c|c}
3 \inp{\ips{\stap}}
4 \const{*Complex} \const{Anch.-IO} \const{Cont.-IO}
5 \cand{\stap} \vio{*!} \vio{} \vio{}
6 \cand{\Optimal}{\sap} \vio{} \vio{} \vio{*}
7 \cand{\tap} \vio{} \vio{*!} \vio{}
8 \end{tableau}
Floats: your turn

(α) Insert your favourite meme \[ \text{\textbackslash includegraphics} \]

(β) Make it into a figure with a caption and label \[ \text{\textbackslash begin\{figure\}} \ldots \text{\textbackslash end\{figure\}} \]

(γ) Cross-reference it in the running text somewhere \[ \text{\textbackslash autoref} \]

(δ) Then pick 1 (depending on applicability)
   - Make a table with horizontally merged cell \[ \text{\textbackslash multicolumn} \]
   - Make an OT tablue with the ot-tableau pack. \[ \text{\textbackslash begin\{tableau\}} \ldots \text{\textbackslash end\{tableau\}} \]
Trees
Trees

There are lots of packages that can make pretty trees - today we’ll be using forest.

forest has a lot of customisation options; as with most packages, the details of its functionality are available in its documentation (hosted on CTAN too, click here).

qtree is another popular option.

Trees can be implemented within figures or examples (e.g. using the expex package).

Unlike images (of trees), these can be changed and customised, and adhere to global formatting (e.g. font).
Trees: the \texttt{forest} package

Preamble: \texttt{\usepackage[linguistics]{forest}} (the linguistics option formats standard syntax trees well).

Environment: \texttt{\begin{forest}...\end{forest}}

Content: \texttt{forest} uses (labelled) square brackets as its input.

\begin{verbatim}
1 \begin{forest} [VP [DP] [V’ [V] [DP]]] \end{forest}
\end{verbatim}
Forest: formatting

You can format within nodes the same way you would normal text - e.g. new lines or italics.

```latex
\begin{forest}
[VP
  [DP
    \textit{Linguists}]
  \textit{like}
  [DP
    \textit{trees}]]
\end{forest}
```
forest: formatting

There are plenty of other formatting options: distance between nodes, alignment, tree direction, line thickness, fonts, colours, borders...

Some example commands:
- `calign = ordinal number` changes which child is aligned with parent
- `grow=direction` changes the direction of the tree
- `draw` draws (customisable) borders around each node
- `inner sep = length` sets size of node
- `s sep = length` changes distance between siblings
- `l sep = length` changes distance between parent and child
- `fill = colour` fills nodes with colour
forest: formatting

If these options are specified within a node, they will only apply to that node.

If you want them to more of the tree, we need to *propagate* them, using the command `for tree={options}`.

If `for tree` goes *directly after* `\begin{forest}`, the options will apply to the whole tree.

Otherwise, you can use `for tree` in a node, and it will apply to that node and all its descendants.
\begin{forest}
  [A tree
    [blue nodes, for tree={fill=blue!30}
      [more[nodes][colour change, fill=green!30]]
      [red nodes, for tree={fill=red!30} [blah][blah][blah]]
    ]
  ]
\end{forest}
forest: empty nodes

Empty nodes are useful but they don’t look very nice when done automatically.

\begin{forest}
[Linguists
    [like
    [trees]]
\end{forest}
forest: empty nodes

We can use nice empty nodes to improve this.

1 \begin{forest} [, nice empty nodes [Linguists] [2 [like] [trees]]] \end{forest}
forest: empty nodes

There are also variants called pretty nice and fairly nice (you may have to copy the code for these separately though).

1 \begin{forest}[,, \textbf{pretty/fairly} nice empty
2 nodes[Linguists][[like][trees]]\end{forest}
To add arrows to trees, we have to label the relevant nodes (using \texttt{name=label}), and then draw a line between them, using \texttt{draw [-->]} (<source>) to[<options>] (<target>).

\begin{forest}
[CP [DP, name=spec CP] [VP [DP ] [V'] [V] [textit{t}, name=object]]]
\draw[-->] (object) to (spec CP);
\end{forest}
We need our arrow to curve - this is when the \texttt{[options]} come in handy.

\begin{verbatim}
1 \begin{forest} [CP [DP,name=spec CP] [VP [DP ] [V' [V] [\textit{t},name=object]]]]
2 \draw[->] (object) to[out=south west, in=south] (spec CP);
\end{forest}
\end{verbatim}
forest: triangles

Triangles are also very easy in forest, using the roof command.

\begin{verbatim}
1 \begin{forest}
2 [VP [DP [Linguists, roof] ]
3 [V' [V \ like ] ] [DP [trees, roof]]
4 \end{forest}
\end{verbatim}
forest: beyond syntax

Syntax trees are far from the only thing you can use forest for.

More generally: flowcharts, decision trees, etc.

Computational linguistics diagrams, e.g. Finite State Automata/Transducers, Push-Down Automota.

Phonology, e.g. Government Phonology (including the GP1 style).

Other linguistics: language families, etymologies, diachronic change representations...
Trees: your turn

Try to recreate the following (or a similar sentence):

```
CP
  /
Did     TP
  /
  /
  you
  /
  /
  /
  t   VP
    /
    /
    /
    like   DP
      /
      /
      /
      the workshop?
```

Note how nice those empty nodes look.

The distance between parents and children is 1.5cm (\texttt{l sep=...}).

The arrow goes both out and in at the southwest of its nodes.

Node options (like names and roofs) go after the node content, separated by commas ([\texttt{node name, option 1, option 2}]).
Special Characters
Special Characters

Special characters are anything that goes beyond what you have on your keyboard.

In this case, we will focus on the IPA, mathmode, and semantics symbols.

And also introduce you to detexify to help you get the code you need for each symbol quicker.
IPA and \texttt{tipa}

The package needed for the IPA is called \texttt{tipa}

Preamble: \texttt{\usepackage{tipa}}

As with other topics seen, this introduces new characters, environments and commands
Why \texttt{tipa}?

A new 256 character encoding for phonetic symbols which includes all the symbols and diacritics found in the recent versions of IPA and some non-IPA symbols.

Easy input method in the IPA environment

Extended macros for accents and diacritics

A flexible system of macros for ‘tone letters’

An optional package (vowel.sty) for drawing vowel diagrams
tipa Characters

Every TIPA phonetic symbol has 3 things:

A unique symbol name, such as Turned A, Hooktop B, Schwa.

A corresponding control sequence, or macro, name, such as `\textturna`, `\texthtb`, `\textschwa`.

And many have a shortcut character that refers to a single character that is assigned to a specific phonetic symbol and that can be directly input by an ordinary keyboard.
Macro characters

The name used as a control sequence is usually an abbreviated form of the corresponding symbol name with a prefix \text...

With some small changes, such as removal of suffixes, ‘l’ and ‘r’ for left and right respectively, sc for small capitals, ht for hooktop, ct for curly tail

<table>
<thead>
<tr>
<th>Symbol name</th>
<th>Macro name</th>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turned A</td>
<td>\textturna</td>
<td>e</td>
</tr>
<tr>
<td>Glottal Stop</td>
<td>\textglotstop</td>
<td>?</td>
</tr>
<tr>
<td>Right-tail D</td>
<td>\textrtaid</td>
<td>Ʉ</td>
</tr>
<tr>
<td>Small Capital G</td>
<td>\textscg</td>
<td>ɕ</td>
</tr>
<tr>
<td>Hooktop B</td>
<td>\texthtb</td>
<td>ə</td>
</tr>
<tr>
<td>Curly-tail C</td>
<td>\textctc</td>
<td>ɕ</td>
</tr>
<tr>
<td>Crossed H</td>
<td>\textcrh</td>
<td>ℱ</td>
</tr>
<tr>
<td>Beta</td>
<td>\textbeta</td>
<td>β</td>
</tr>
</tbody>
</table>
Shortcut characters

Shortcut  :  ;  ”
IPA      :  ‘  ’
Shortcut  0 1 2 3 4 5 6 7 8 9
IPA      ı  ‿  ş  ş  ş  ş  ş  ş  ş  ş
Shortcut  @ A B C D E F G H I
IPA      ə  a  β  θ  η  θ  θ  θ  θ
Shortcut  J K L M N O P Q R S
IPA      ɹ  ɹ  ɹ  ɹ  ɹ  ɹ  ɹ  ɹ  ɹ
Shortcut  T U V X Y Z —
IPA      θ  υ  υ  χ  χ  χ  χ  χ  χ
Diacritics and Tone

Almost infinite possibilities, too many for a single slide, so will highlight some.

<table>
<thead>
<tr>
<th>Macro</th>
<th>IPA environment</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accents</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'a</td>
<td>'a</td>
<td>á</td>
</tr>
<tr>
<td>&quot;a</td>
<td>&quot;a</td>
<td>ä</td>
</tr>
</tbody>
</table>

| **Suprasegmentals** |                 |        |
| textsubbridge{t}   | —[t             | ť      |
| textsubplus{o}     | —+o            | polator |

| **Tone (\usetone{tipa})** |                 |        |
| tone{55}ma             | tone{55}ma     | ℓma    |
| tone{214}ma            | tone{214}ma    | ∨ma    |
Using \texttt{tipa}

There are two ways to input phonetic symbols using \texttt{tipa}, and the first is inputting macro names or shortcut characters within special environments

\texttt{Environment : \begin{IPA} \ldots \end{IPA} OR \texttt{textipa}\{\ldots\}}

In these environments you can use the shortcut symbols

\begin{verbatim}
1 \texttt{textipa \{""Ekspl@""neIS@n\}}
2 \texttt{or}
3 \texttt{\begin{IPA}
4 \quad \texttt{[""Ekspl@""neIS@n]
5 \end{IPA}}
\end{verbatim}

\[\text{[\text{\textedsl\'nei\textedsl\'en}] or [\text{\textedsl\'nei\textedsl\'en}]}\]
Using \texttt{tipa}

The second way is to enter the macro commands in the normal text environment.

\begin{verbatim}
1 [\textsecstress \textepsilon kspl\textschwa
2 \textprimstress ne\textsci \textesh \textschwa n]
\end{verbatim}

[ˌɛkspləˈniʃən]
How on earth am I going to remember all of these characters?

Luckily, you don’t! All are easily googleable, and many websites have special cheatsheets. For example:


https://ptmartins.info/tex/tipacheatsheet.pdf

BUT there are clear and easy to remember patterns! e.g. if you know what `\textrtailed` (right tailed-d) does, you can probably guess how to get a t with a right tail, or a n with a left tail... that is a huge advantage over ALT codes or word insert feature
Detexify

Another way of accessing the code for all of the IPA symbols, and all the special characters more generally is through Detexify!

All are easily googleable, and many websites have special cheatsheets
Mathmode

Good for any equations you may need to use, and also good for many semantic symbols

You can use any of these ‘delimiters’ to typeset your math in inline mode

\$...\$

\begin{math}...\end{math}

Can also have a separate equation on separate line that you can refer back to later using the \autoref from earlier!

\begin{equation}...\end{equation}
Mathmode Characters

Again, too many to mention here, but I can refer you to a really good summary on these websites:


https://www.overleaf.com/learn/latex/List_of_Greek_letters_and_math_symbols

Small sample to get you started:
Mathmode Characters

1 $\Delta, \lambda$
2 $a \pm b$
3 $\sigma$
4 $\sum_{i=1}^{n} x_i$
5 $a \leq b$
Semantics characters

Most can be done in mathmode, since the majority of semantics symbols are just maths symbols:

\neg \quad \neg \quad \in \quad \in \quad \& \quad \& \quad \subset \quad \supset \quad \subset \supset \quad \rightarrow \quad \rightarrow \quad \alpha \quad \alpha

\exists \quad \exists \quad \lambda \quad \lambda \quad \forall \quad \forall \quad \phi \quad \phi

1 $\lambda x.\ $ Different_from($x$,Sam)

$\lambda x.\ $ Different_from($x$,Sam)
Semantics characters

There are a few things you do need other packages or methods for, namely DRT and box/diamond notation.

DRT can be done in a tabular environment or array.

For box/diamond notation, you’ll need the \texttt{latexsym} package, and many other less frequent symbols will require the same package:

\begin{verbatim}
1 \usepackage{latexsym} ← In the preamble!
2 \diamond
\end{verbatim}
Great big list


Loads of different characters we haven’t had the time to go over!
Special characters: your turn

Can you write your name in IPA using latex?

Can you write Einstein’s famous mass-energy equivalence equation?
Glossing
Glossing

There are many packages that can be used for glossing. The one we will look at today is the \texttt{expex} package.

\texttt{expex} has many customisation options for numbering, spacing, text style, and so on, which are all explained in the package documentation.

Another popular package is \texttt{gb4e}. It has fewer customisable options but works fine for most data.
expex package

Preamble: `\usepackage{expex}

expex introduces new environments as well as new commands.

Remember, environments must be opened and closed.

Environment 1: `\ex ... \xe`

1 `\ex An example \xe`

(5) An example
expex package

Environment 2: \pex ... \xe

1 \pex An example with parts
2 \a Part a
3 \a Part b
4 \xe

(6) An example with parts
   a. Part a
   b. Part b
Glossing using `expex`

`\begingl (this is an environment which needs to be closed)`

`\gla <language line> //`

`\glb <gloss line> //`

`\glc <another gloss line> //`

`\glft <translation> //`

`\endgl (close environment)`
Glossing using \texttt{expex}

egin{verbatim}
1 \ex
2 \begingl
3 \gla Yeh Urdu he //
4 \glb This Urdu is //
5 \glft 'This is Urdu.' //
6 \endgl
7 \xe
\end{verbatim}

(7) Yeh Urdu he.
    This Urdu is
    'This is Urdu.'
Glossing using `expex`

1 \ex
2 \begingl
3 \gla Yeh Urdu he/
4 \glb Subj Obj V/
5 \glc This Urdu is/
6 \glft ‘This is Urdu.’/
7 \endgl
8 \xe

(8) Yeh Urdu he.
   Subj Obj V
   This Urdu is
   ‘This is Urdu.’
Glossing using `expex`

1 \texttt{\textbackslash ex \textbackslash ljudge*}
2 \texttt{\textbackslash begingl}
3 \texttt{\textbackslash gla Yeh nahi Urdu he} //
4 \texttt{\textbackslash glb This not Urdu is\texttt{\textbackslash textsc{.3sg}}} //
5 \texttt{\textbackslash glft \textquoteleft This is not Urdu.	extquoteright //}
6 \texttt{\textbackslash endgl}
7 \texttt{\textbackslash xe}

(9)*Yeh nahi Urdu he.
   This not Urdu is.3SG
   ‘This is not Urdu.’
Glossing: your turn

Gloss your own language examples in \LaTeX. Try to produce a multi-part example.
The COPiL template
Useful links

**CTAN**: repository for packages (and their documentation).

**Overleaf documentation**: comprehensive help for everything relating to Overleaf.

**StackExchange**: Q+A forum.

**BibTeX website**: guides and tips to using BibTeX.

**Tables Generator**: create (or import from Excel) tables.

**Detexify**: draw symbols to find their code/necessary packages.

Character cheat-sheets for **TIPA** and for **mathmode symbols**.

**COPiL submission page**: guidelines for submission and template.
End