

# Impromptu L<sup>A</sup>T<sub>E</sub>X workshop at the 2015 Chicago LSA Institute<sup>\*</sup>

Adam Liter  
[adam.liter@gmail.com](mailto:adam.liter@gmail.com)

July 22, 2015

## 1 Terminology

This section is largely a brief recap of §1 of Alan Munn’s *A Beginner’s Guide to L<sup>A</sup>T<sub>E</sub>X (on the Mac)*.<sup>1</sup> But you should really just read that whole PDF (and ignore the parts specific to Mac if you’re not on a Mac). It is both short and useful.

**T<sub>E</sub>X Distribution** Contains all of the programs and packages that will be used to process and compile your `.tex` file. There are two main distributions: TeX Live and MiKTeX. There is also a distribution called MacTeX, which is a wrapper around TeX Live that does some stuff to make it work nicely on a Mac. MiKTeX is for Windows only, and it is not based on TeX Live.<sup>2</sup>

**Engines** There are a few different engines that are standardly used to process a `.tex` file and turn it into a PDF, including pdfL<sup>A</sup>T<sub>E</sub>X, X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X, and LuaL<sup>A</sup>T<sub>E</sub>X.

**Editor** The application that is used to write the `.tex` file. See [here](#) for a long list of editors to choose from.

**Previewer** An application for viewing the output of compiling the `.tex` file with an engine. Many editors integrate a previewer into the editor.

**Compiling** The act of processing a `.tex` file with an engine to produce (most likely) a PDF. Can sometimes loosely be used interchangeably with “typesetting”.

**Preamble** Refers to the part of the document between `\documentclass` and `\begin{document}`. It is where you can load packages and define new commands, among other things. See Figure 1.

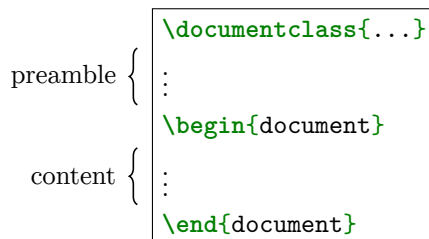


Figure 1: Schematic structure of a L<sup>A</sup>T<sub>E</sub>X document

---

<sup>\*</sup>The `.tex` file that produced this PDF is available at <https://github.com/adamliter/latex-workshop>. There are a few dependencies and oddities that might prevent you from actually being able to compile it yourself on your machine. (It definitely won’t compile on one of the online editors because you need to enable shell escape.) Anyway, it’s not worth explaining these things any further in a footnote. This document already has enough footnotes as it is, but you’re welcome to look at the source code of the document to get an idea of how to write something in L<sup>A</sup>T<sub>E</sub>X.

<sup>1</sup>A PDF of this is available at <https://www.msu.edu/~amunn/latex/nano-companion.pdf>. If you have trouble accessing it, try refreshing the page a few times. The web servers at Michigan State University can sometimes be sort of shitty.

<sup>2</sup>For discussion of the differences between MiKTeX and TeX Live, see <http://tex.stackexchange.com/q/20036/32888>. If you’re on Linux, do *not* install TeX Live via your package manager!!! You should instead install a “vanilla” version of TeX Live. See <http://tex.stackexchange.com/q/1092/32888>. See also §2.1 of this handout.

**TeX.SX** Throughout this document, you will probably see numerous references to TeX.SX. This is short for [TeX Stack Exchange](#). If you're not familiar with the [family of Stack Exchange websites](#), you should really check them out. Each site is a Q&A website for a specific topic, but the sites are intended to be repositories of knowledge in addition to Q&A sites, so they aren't like your normal web forum. Most sites go with the suffix of .SE, but the folks that use TeX Stack Exchange are a bit idiosyncratic and generally prefer the suffix .SX.

## 2 Setting up your machine

### 2.1 Installing a T<sub>E</sub>X distribution

There are two relatively new and popular web editors for L<sup>A</sup>T<sub>E</sub>X—namely, [ShareLaTeX](#) and [Overleaf](#). In this workshop, we will use these web editors and not bother with having folks install their own T<sub>E</sub>X distribution.

However, I still wanted to cover it in the handout. Feel free to read through this on your own time.

#### 2.1.1 Mac

If you're on a Mac, you should install [MacT<sub>E</sub>X](#). MacT<sub>E</sub>X is all TeX Live underneath with just a thin wrapper that makes things work smoothly on a Mac. MacT<sub>E</sub>X also installs two editors—TeXShop and TeXworks—and a program for managing a .bib file, called BibDesk.

#### 2.1.2 Linux

Do *not* install TeX Live on Linux via your package manager! The T<sub>E</sub>X distribution that you will get from your package manager will most likely be out of date, which will preclude you from being able to update packages.

Instead, you should [install a “vanilla” version of TeX Live](#).

#### 2.1.3 Windows

The easiest thing to install on Windows is MiK<sub>T</sub>TeX,<sup>3</sup> which is a different distribution than TeX Live. MiK<sub>T</sub>TeX doesn't install every package but instead installs a minimal distribution and allows you to install packages on the fly when compiling your document if the requisite package is not already installed.

At one point, there were security concerns about MiK<sub>T</sub>TeX and thus it was preferable to install TeX Live. However, these security concerns seem to have been mitigated, and it's not clear that there is a huge reason to prefer a TeX Live installation on Windows.<sup>4</sup> Moreover, it is not as straightforward to install TeX Live as it is to install MiK<sub>T</sub>TeX. Nonetheless, if you wish to do so, see [here](#).

### 2.2 Keeping your T<sub>E</sub>X distribution up to date

It is good practice to periodically update your T<sub>E</sub>X distribution. A T<sub>E</sub>X distribution includes a bunch of packages, which are periodically edited by their maintainers. These packages are hosted on the [Comprehensive T<sub>E</sub>X Archive Network \(CTAN\)](#). You should thus periodically update things in case the maintainers of packages find a bug and fix that bug or in case they add new features to the package.<sup>5</sup>

In addition to periodically updating the packages, you will also want to periodically update the entire distribution. Just like with packages, new features are developed or bugfixes are sometimes made to the engines themselves and other binaries that are the core of a T<sub>E</sub>X distribution.

---

<sup>3</sup>Disclaimer: I know very little about Windows and MiK<sub>T</sub>TeX.

<sup>4</sup>See [this question and its answers on TeX.SX](#) for discussion.

<sup>5</sup>One thing that is also great about L<sup>A</sup>T<sub>E</sub>X, in stark contradistinction to Word, is its backward compatibility. That is, even if package authors introduce new features, they will make sure that any document you previously typeset using their package will be something that you can still typeset using the new updated version of their package. If package authors do decide to break backwards compatibility, they will usually create a new package with an entirely new name, which effectively maintains backward compatibility because the old package will always be available for use. On the other hand, with Word, you're lucky if you can open a file from last year's version of Word with this year's version of Word, much less have the formatting look even remotely the same. With L<sup>A</sup>T<sub>E</sub>X, you could typeset a L<sup>A</sup>T<sub>E</sub>X file written in 1739 and the output you get would be identical to the output you got in 1739.

For TeX Live, there is a new distribution that is released every year. The current one is TeX Live 2015.

When the new distribution is about to be released, the old one is “frozen”. Once it is frozen, you will no longer be able to update packages, so you will want to install the newest version of TeX Live for any new features or bugfixes to the engines and other binaries as well as for the ability to continue to periodically update packages.

## 2.3 Local files

One thing you will presumably also want to do at some point is set up a directory for local files that you want to be accessible to all of your `.tex` files, regardless of where that `.tex` file is actually stored on your machine.

The most obvious use case for such a directory is for the purposes of maintaining a single master bibliography file on your computer that can be used for citations in all of your `.tex` files (see §3.13).

Where and how to set up this directory depends on your distribution, TeX Live or MiKTeX. What is common to both cases, however, is that the directory must conform to the standard T<sub>E</sub>X Directory Structure (TDS) hierarchy. A minimal example of a directory structure that conforms to this standard is given in Figure 2.<sup>6</sup>

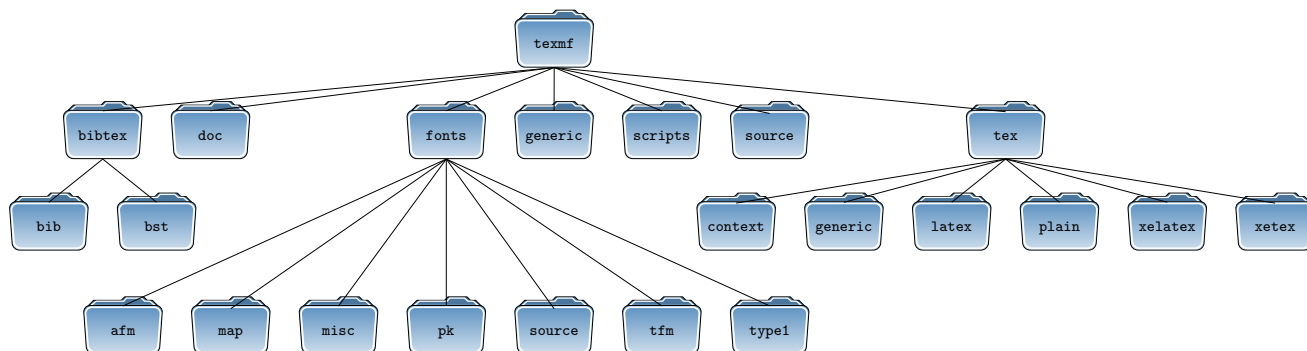


Figure 2: A minimal directory that conforms to the TDS standard

It is necessary to conform to this standard so that the engine you use to compile your `.tex` file can find certain types of files. For example, if you maintain a single master `.bib` file, it should be placed in the folder `texmf/bibtex/bib`. If you put it in any other folder, the engine you use to compile your document will not find it because it is only programmed to look for bibliography files inside the `texmf/bibtex/bib` folder.

In what follows, I describe how to set up a local TDS-compliant directory for both TeX Live and MiKTeX. For further discussion, see [this question and its answers on TeX.SX](#).

### 2.3.1 TeX Live

In TeX Live, engines are set up to look in certain places for files that your `.tex` file might depend on. TeX Live specifically provides two places for users to put their own files, such as style files or bibliography files. These two places are identified by their variable names, `TEXMFHOME` and `TEXMFLOCAL`.

You can change the values of these variables. `TEXMFHOME` usually refers to the path `~/Library/texmf` on Mac, the path `~/texmf` on Linux, and the path `C:\Users\<user name>\texmf` on Windows, but you could change the variable `TEXMFHOME` to point to `~/Dropbox/texmf` instead, if you wanted to.

`TEXMFHOME` and `TEXMFLOCAL` have the same semantics; that is to say, they are both places where users can put their own files that are not part of TeX Live. However, `TEXMFLOCAL` will be overwritten every time you install a new version of TeX Live. For this reason, it is probably best to keep all of your local files in `TEXMFHOME`.<sup>7</sup>

As mentioned above, the default value for `TEXMFHOME` on a Mac is `~/Library/texmf`, the default value on Linux is `~/texmf`, and the default value on Windows is `C:\Users\<user name>\texmf`.<sup>8</sup>

<sup>6</sup>There are even more folders in a maximal TDS directory, but the ones depicted in Figure 2 are probably enough for most use cases. If you’re interested in reading more about TDS, you can do so at <https://www.tug.org/tds/tds.pdf>.

<sup>7</sup>For discussion, see <http://www.tex.ac.uk/FAQ-what-TDS.html>.

<sup>8</sup>If you’re still on Windows XP, it should be `C:\Documents and Settings\<user name>\texmf` instead of `C:\Users\<user name>\texmf`.

If you are unsure what the value of `TEXMFHOME` is, you can check it by going to the command line and running `kpsewhich -var-value TEXMFHOME`.

If, for example, you're on a Mac and haven't changed the default setting, this should return the following file path: `/Users/<user name>/Library/texmf`.<sup>9</sup>

Even though the variable `TEXMFHOME` has a value, the folder might not exist. In order to make use of the ability to maintain your own personal bibliographic and style files, you need to make a TDS-compliant directory at the location of `TEXMFHOME`.

If you're on a Mac, you're in luck, because Alan Munn has written an [app that will do this automatically for you](#).

If you're on Linux, you can do what is shown in Listing 1.<sup>10</sup>

```
mkdir -p $(kpsewhich -var-value TEXMFHOME)/{doc,generic,scripts,source}
mkdir -p $(kpsewhich -var-value TEXMFHOME)/bibtex/{bib,bst}
mkdir -p $(kpsewhich -var-value TEXMFHOME)/fonts/{afm,map,misc,pk,source,tfm,type1}
mkdir -p $(kpsewhich -var-value TEXMFHOME)/tex/{context,generic,latex,plain,xelatex,xetex}
```

Listing 1: Make a minimal TDS-compliant directory at `TEXMFHOME`

I don't know anything about the Windows Command Prompt, so the only thing I can suggest is creating the folders manually using File Explorer.

### 2.3.2 MiKTeX

MiKTeX is different from TeX Live in that it allows users to select directories to be used for storing local files through a graphical user interface, rather than having an environment variable that maps to such a directory.

In order to get things set up on MiKTeX, you will want to first set up a TDS-compliant directory somewhere on your computer. MiKTeX recommends making the directory at `C:\Local TeX Files`. Once you create the folder `Local TeX Files` in the `C:\` directory, then you will need to create the folders shown in Figure 2.<sup>11</sup>

After you have created all of these folders, you can follow [these steps](#).

## 3 General L<sup>A</sup>T<sub>E</sub>X stuff

### 3.1 L<sup>A</sup>T<sub>E</sub>X philosophy

This subsection is very similar to §3 of Alan Munn's *A Beginner's Guide to L<sup>A</sup>T<sub>E</sub>X (on the Mac)*.

L<sup>A</sup>T<sub>E</sub>X was designed with the intent of separating content from formatting. This is quite different from a what-you-see-is-what-you-get (WYSIWYG) editor like Word, where you see the output of your content formatted as you go along.

Something that goes hand in hand with separating content from formatting is that formatting should be given a semantics. What does this mean? (Bahhhh duhhhh chhhh!) This means that if you want L<sup>A</sup>T<sub>E</sub>X to format things that are similar in nature in the same way, then you should give them the same semantic meaning.

Alan gives the example of section headings. The proper way to make a section heading in L<sup>A</sup>T<sub>E</sub>X is to write it like `\section{Section title}`. The command `\section{}` gives the content "Section title" the semantics of being a **section**. Then, if you want to change anything about how your sections are formatted, you can change this in the preamble of your document. This contrasts with how many folks use Word where they would change this for each

<sup>9</sup>The '~' is used as shorthand for a user's home directory. That is to say, '~/`Library/texmf`' is the same as '`/Users/<user name>/Library/texmf`' on a Mac.

<sup>10</sup>This will also work on a Mac. Note, however, that it will only work after TeX Live has been installed, because the command line tool `kpsewhich` is part of TeX Live.

<sup>11</sup>Note that in this case the folder called `Local TeX Files` is the same as the `texmf` folder depicted in Figure 2. You should *not* put a folder called `texmf` inside of `Local TeX Files`. Instead, treat `Local TeX Files` as the `texmf` folder.

individual section heading.<sup>12</sup> For example, if you want the number preceding all of your section headings to be blue, you can change this at the beginning of the document, like in Listing 2.

```
\documentclass{article}

\usepackage{color} % this package provides the command \color{}
\renewcommand\thesection{\color{blue}\arabic{section}}

\begin{document}

\section{Introduction}

Blah blah.

\section{Experiment}

Blah blah.

\section{Conclusion}

Blah blah.

\end{document}
```

Listing 2: Example of semantic markup in L<sup>A</sup>T<sub>E</sub>X for section headings

## 3.2 Titles

To typeset a title, an author, and a date in a paper using the basic `article` class, you can do what is shown in Listing 3.

```
\documentclass{article}

\title{Super awesome title}
\author{Best Author}
\date{July 22, 2015}

\begin{document}

\maketitle

\section{Introduction}

Blah blah.

\end{document}
```

Listing 3: A basic example of how to typeset the title of a paper using the `article` class

---

<sup>12</sup>Note that Word also allows for semantic markup despite the fact that most people do not use it. If you cannot convince your Word-using friends and family to switch to L<sup>A</sup>T<sub>E</sub>X, you should at least try to get them to use semantic markup in Word if they don't already do so. :)

### 3.3 Quotes and dashes

One idiosyncrasy of  $\text{\LaTeX}$  that you will have to get used to is how to typeset quotes and dashes.

To typeset double open quotes, write ````.

To typeset double close quotes, write `''`.

To typeset a single open quote, write ```.

To typeset a single close quote, write `'`.

To typeset an en-dash, write `--`.

To typeset an em-dash, write `---`.

Note, however, that if you process your file with an engine that plays nicely with UTF-8 encoded documents (see §4.1), you can enter these characters directly into your editor.

Doing this has the advantage of making your documents more readable, but note that it *only* works if you use UTF-8 encoding and a compatible engine. Also, it is good to know about the old way of typesetting these ligatures using  $\text{\LaTeX}$  since you will probably see many instances of this on the internet.

### 3.4 Formatting text

To typeset something in bold, use `\textbf{}`.

To typeset something in italics, use `\textit{}`.

To typeset something in small caps, use `\textsc{}`.

To typeset something in a mono-spaced font, use `\texttt{}`.

To underline something, don't.<sup>13</sup>

One important thing to note about all of the foregoing commands is that using them directly in your document is *not* good practice. They are not semantic commands. Let's consider an example. Being a linguist, you will probably want to typeset certain glossed features in small caps, like NOM, for instance. You might think to write `\textsc{nom}`.

This is *bad practice*. Instead, you should give glossed feature abbreviations like this a semantics, since you will presumably want to typeset them all the same way. To do this, you could declare `\newcommand*{\Fts}[1]{\textsc{#1}}` in your preamble. Then, you would be able to write `\Fts{nom}` instead of `\textsc{nom}`.

A further example can be seen if you look at the source code for this handout. You will notice that I typeset all the names of packages in a mono-spaced font. Rather than writing, for example, `\texttt{forest}`, I have written `\Package{forest}`. In my preamble, I defined the `\Package{}` command in the following way: `\newcommand*{\Package}[1]{\texttt{#1}}`. If I ever wanted to change how the name of every package is typeset in this document, I would only need to change it once in my preamble.

### 3.5 Footnotes

To typeset footnotes, use `\footnote{}`.

### 3.6 Special characters

There are several characters that are treated as special characters in  $\text{\LaTeX}$ . These are `#` `$` `%` `&` `~` `_` `^` `\` `{` `}`.

If you ever want to print any of these characters in the output, you need to escape them with `\`.<sup>14</sup>

The character `#` is used for passing arguments to macros.

---

<sup>13</sup>Underlining is really frowned upon in the typography community. I also personally do not like it. However, if you have a really, really, really, really, really (really) good reason, then I suppose you can use `\uline` from the `ulem` package. When you load, `ulem` be sure to pass it the optional argument `normalem`.

<sup>14</sup>One exception to this is the escape character itself, `\`, because the sequence `\\` has a special meaning in  $\text{\LaTeX}$ , used for line breaks in tables. If you wish to render the character `\` you can use the command `\textbackslash`. Two further exceptions are `^` and `~`. Preceding these two characters with `\` is used for appending diacritics in  $\text{\LaTeX}$  (though see §4.1 for a better way of doing this). If you wish to print these characters, you will need to do `\^{}{}` and `\~{}{}`, respectively (though see [this post on TeX.SX](#) for suggestions of better ways to typeset a tilde).

The character ‘\$’ is used for entering math mode (see §3.7).

The character ‘%’ is used for writing comments in the source document (cf. Listing 2).

The character ‘&’ is used for separating columns in a table (see §3.8).

The character ‘~’ is a non-breaking space.

The character ‘\_’ is used for subscripts in math mode.

The character ‘^’ is used for superscripts in math mode.

The character ‘\’ is the escape character.

The characters ‘{ }’ are used for delimiting the arguments to commands.

### 3.7 Math mode

One thing worth knowing about L<sup>A</sup>T<sub>E</sub>X is that it has a distinct mode for typesetting math, creatively called math mode. There is inline math, triggered by `$...$`, and display math, triggered by `\[...\]`.

For linguists, math mode is something that is mostly useful for typesetting semantics.<sup>15</sup> For example, `$\lambda x$` will produce  $\lambda x$ .

### 3.8 Tables

Tables are admittedly a bit of a pain in the ass in L<sup>A</sup>T<sub>E</sub>X. Typesetting them takes a while to get used to. Let’s see an example of a basic table, such as the one in Listing 4.

```
\documentclass{article}
\begin{document}

\begin{tabular}{lcr}
  Left-aligned column & Center-aligned column & Right-aligned column \\
  56\% & 75\% & 34\% \\
\end{tabular}

\end{document}
```

Listing 4: A basic table in L<sup>A</sup>T<sub>E</sub>X

The code in Listing 4 will produce the following table.

Left-aligned column	Center-aligned column	Right-aligned column
56%	75%	34%

One useful package for making aesthetically pleasing tables is the package called `booktabs`. It provides commands called `\toprule`, `\bottomrule`, and `\midrule` for nicer horizontal rules in a table.

Consider Listing 5, which produces the following output.

	Passive sentences	Active sentences
Adults	99%	98%
Children	56%	87%

### 3.9 Images

One thing you will often want to do is include images in a document. This is what the package `graphicx` is for. Let’s look at the example in Listing 6.

<sup>15</sup>This is a bit of an overstatement. There are other use cases. Typesetting feature bundles in morphosyntax is one such use case, for example.

```

\documentclass{article}
\usepackage{booktabs}
\begin{document}

\begin{tabular}{lll}
\toprule
& Passive sentences & Active sentences \\ \midrule
Adults & 99\% & 98\% \\
Children & 56\% & 87\% \\
\bottomrule
\end{tabular}

\end{document}

```

Listing 5: A table in L<sup>A</sup>T<sub>E</sub>X using the package booktabs

```

\documentclass{article}
\usepackage{graphicx}
\graphicspath{ {figure/} }
\begin{document}

\includegraphics[width=.8\textwidth]{super-important-graph}

\end{document}

```

Listing 6: An example of including an image in a document

Notice that in the preamble of the document, we declared `\graphicspath{ {figure/} }`. This means that the package `graphicx` will look in the folder called `figure` for images.<sup>16</sup> So, in order to get this `.tex` file to compile, you would want to save it in a folder; then, in that same folder, you would want to create a new folder called `figure`. Inside that folder, you should put the file `super-important-graph.pdf`.<sup>17</sup> Notice that the file extension is omitted in the call to `\includegraphics{}`. This is a best practice because it allows `graphicx` to try a bunch of different file extensions.

Notice also that we passed an optional argument to `\includegraphics{}` in the form of a *key val list*. The key **width** can take a value that specifies what the width of the image that appears in the typeset document should be. You could give it a value of `6in` if you wanted, but it is often much more sensible to specify the width in terms of a dynamically defined value. In this case, the super important graph will always occupy 80% of the space allocated to the text, even if we change the margins of the document.

### 3.10 Captioning and numbering

Typesetting a table and including an image is great and all, but we want to be able to caption and number them.

However, for the purposes of an introductory workshop, you should really just know what math mode is.

<sup>16</sup>You don't need to do this. If you prefer, you can just put the image file in the same folder as the `.tex` file. The working directory (*i.e.*, the folder that the `.tex` file is in) is a place that the engine will always search when typesetting a document. So if you're struggling with setting up a local `texmf` folder as described in §2.3, you can always just put files in the same folder as your `.tex` file for the time being. But anyway, a reason you might want to have a separate folder dedicated for figures is to avoid clutter. It's really up to you.

<sup>17</sup>The package `graphicx` doesn't always play nicely with spaces and underscores in file names, so you should avoid using those things in the names of your image files.



### 3.10.1 Floats

One common way to do this is to use floats. In addition to automatically numbering tables and figures, floats also allow us to provide a caption. One thing to know about floats is that  $\text{\LaTeX}$  has a special way of handling how they are typeset. Suffice it to say, it's rather complicated.<sup>18</sup> All you need to know is that  $\text{\LaTeX}$  has a special algorithm for placing floats in the best possible spot, according to general typographical standards.

These places are usually one of four places: right where they are written in the source document, the top of a page, the bottom of a page, or on their own separate page. These four places correspond to four optional arguments that you can pass to a float environment, `htbp`, respectively.

It is generally best practice to pass all four options to a float, at least initially. Only when you finish writing the document should you fiddle with the placement of floats if you think  $\text{\LaTeX}$ 's algorithm has not done a good job. However, while you're writing a document, leave all four options and let  $\text{\LaTeX}$  decide where floats should be placed.

If you would prefer to increase the likelihood that the float will show up in exactly the location that it is specified in the source `.tex` file, you can place a `!` after the `h`.

The command `\caption{}` allows you to give a caption to the table or figure.  $\text{\LaTeX}$  will automatically number the tables and figures in the correct order, so you don't have to worry about that. Semantic markup FTW!

Take a look at two examples in Listing 7. Try typesetting this yourself and see what the result is.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{booktabs}
\begin{document}

This is a table that is a float.

\begin{table}[htbp]
  \centering
  \begin{tabular}{lll}
    \toprule
      & & & Passive sentences & Active sentences & \\ \midrule
    Adults & 99\% & & 98\% & & \\
    Children & 56\% & & 87\% & & \\
    \bottomrule
  \end{tabular}
  \caption{Adult performance compared to child performance}
\end{table}

It might not actually show up in between these two sentences.

This is a figure that is a float.

\begin{figure}[htbp]
  \centering
  \includegraphics[width=.8\textwidth]{example-image-a}
  \caption{Super scientificy graphy thingy}
\end{figure}

It might not actually show up in between these two sentences.

\end{document}
```

Listing 7: Examples of floats in  $\text{\LaTeX}$

<sup>18</sup>You can read more about it [here](#) if you're interested.

### 3.10.2 Non-float options

It is worth mentioning some non-float options for tables and images. One of the reasons that it is worth mentioning these options is because there is a common misconception that tables must go inside `table` environments and images must go inside `figure` environments. This is not true.

When you're writing a paper, it is best to use floats because  $\text{\LaTeX}$  will use its algorithm to place the floats in the best environment. However, it is not always appropriate to use floats.

One example of when you probably don't want to use floats is when you're making a handout. In a handout, you usually want the image or the table to show up exactly where you place the code in the source document.

Nonetheless, in this case you might still want to be able to number and caption the figure or table. The package `capt-of` allows you to do this. An example is given in Listing 8. Note that you will want to put things inside of an enclosing group, such as `\begin{center}...\end{center}`.

```

\documentclass{article}
\usepackage{graphicx}
\usepackage{booktabs}
\usepackage{capt-of}
\begin{document}

This is a table that is not a float.

\begin{center}
\begin{tabular}{lll}
\toprule
& Passive sentences & Active sentences \\
Adults & 99\% & 98\% \\
Children & 56\% & 87\% \\
\bottomrule
\end{tabular}
\captionof{table}{Adult performance compared to child performance}
\end{center}

It will show up in between these two sentences no matter what.

This is a figure that is not a float.

\begin{center}
\includegraphics[width=.8\textwidth]{example-image-a}
\captionof{figure}{Super scientificy graphy thingy}
\end{center}

It will show up in between these two sentences no matter what.

\end{document}

```

Listing 8: Examples of tables and images as non-floats

Another case where you might want to not put a table inside of a float is when you're making a table for a particular morphological paradigm. In linguistics, we usually number such tables just like we number other examples. In §4.2, we will see how to make numbered examples. You can put a `tabular` environment directly inside such an example.

### 3.11 Cross referencing

So automagically numbered tables and images are great and all, but how do I refer to those things in my document? One thing that is great about  $\LaTeX$  is that you can give things `\label`s and `\refer` to them automagically as well. Consider Listing 9.

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}

As can be seen in Figure~\ref{fig:important-graph}, the results clearly show that I'm right.

\begin{figure}[htbp]
  \centering
  \includegraphics[width=.8\textwidth]{example-image-a}
  \caption{Super scientificy graphy thingy}
  \label{fig:important-graph}
\end{figure}

\end{document}
```

Listing 9: An example of referencing a figure in  $\LaTeX$

There are a few things to say about cross-referencing. First, and of particularly important note is the fact that the `\label` comes after the `\caption`. If you try putting the `\label` first, you will get the wrong number, because the command `\caption` is what gives the float its number.

Second, you will notice that I've put a non-breaking space between Figure and `\ref`. This is generally a good practice because it prevents the number from being separated from the description of what it is.<sup>19</sup>

Third, you will also notice that I've given the label a prefix of `fig:`. This isn't strictly necessary, but it is good practice. Imagine that you had a table and a figure. The table contains specific values, and the figure is a graph of those values. It's the same data, so you presumably want to give them similar names. If you use prefixes like this, you could do `\label{tab:super-important-results}` for the table and `\label{fig:super-important-results}` for the graph.<sup>20</sup>

Fourth, and most importantly, is that you must compile your document twice in order for this to work.  $\LaTeX$  does its automagic by first processing the file and automagically generating all of the table and figure numbers for each float. Remember that you never manually gave each float a number, so  $\LaTeX$  has to figure this out. On the second compilation, it inserts the automagically generated numbers into the places where you `\referenced` them. If something went wrong, you will see question marks instead of a number. This most likely means you either only compiled your document once or you have tried to refer to something that you never actually labeled. The most common example of this latter reason is just a simple misspelling of the label that you gave to whatever it is that you're trying to reference.

### 3.12 Those annoying files

One thing you will quickly notice when you typeset a `.tex` file is that a lot of extra files are generated. People tend to initially find this annoying, but it is all of these extra files that allow  $\LaTeX$  to do its magic. For example, the auxiliary file (`.aux`) is integral for cross referencing. Without it, cross referencing just would not work.

To avoid clutter, it's often a good idea to make a new, self-contained folder for each document that you typeset.

One other thing worth noting about all of these extra files is that they sometimes lead to compilation errors. If you introduced an error in your document and you tried typesetting it, it's possible that the extra files got messed up. So, if you tried typesetting your document, received a compilation error, figured out what caused the error, and

<sup>19</sup>If you really like automagic, you might want to check out the `cleveref` package.

<sup>20</sup>Spaces are not allowed in the names of `\label`s.

you’re like 110% sure that you fixed the problem in your `.tex` file, but you’re still getting a compilation error, try deleting all of these extra files and compiling the document again.

### 3.13 Bibliographies

Getting a bibliography to work with  $\LaTeX$  is often one of the big hurdles of learning  $\LaTeX$ , but once you’ve figured it out, it’s really, really, really frikken nice.

**Promissory note** Unfortunately, getting this set up properly requires going into how to install a  $\TeX$  distribution on your computer and how to properly set up a local TDS-compliant directory. If folks are interested, we could do a follow up workshop next Wednesday that goes into how to do this.

### 3.14 Paragraphs

Let’s end this section with paragraphs. Let’s do this for two reasons. First, because semantic markup is awesome. And second, because a very common *really bad practice* of  $\LaTeX$  beginners is to insert line breaks all over the place. Recall from §3.1 that  $\LaTeX$  is all about semantic markup. This goes for paragraphs, too. People who are used to Word are used to pressing ENTER on the keyboard once in order to separate paragraphs. Perhaps unsurprisingly then, many new  $\LaTeX$  users will often do stuff like what is shown in Listing 10.

```
\documentclass{article}
\begin{document}
This is my first awesome paragraph.\\
This is my second paragraph, which is infinitely less awesome because of the line break.
\end{document}
```

Listing 10: Really bad practice for separating paragraphs

The command `\\` does a line break, but it does *not* introduce a new paragraph. In other words, the (first part of the) second sentence is false. In  $\LaTeX$ ’s eyes, the sentence “This is my first awesome paragraph.” is in the *same paragraph* as the sentence “This is my second paragraph, which is infinitely less awesome because of the line break.”

Instead, one should use a command for paragraphs so that we can manipulate the semantics of paragraphs in the preamble of the document in the same way that we manipulated the semantics of sections. The command for separating one paragraph from another paragraph is `\par`. This would get really annoying to type in between all of your paragraphs, so, luckily,  $\LaTeX$  treats an empty line as equivalent to `\par`.

Listing 11 exemplifies *good practice* for typesetting paragraphs. This good practice allows us to manipulate the semantics of paragraphs in the preamble, so we can typeset them as we like, without having to modify each individual paragraph.

There’s generally no need to modify the default semantics for paragraphs. However, in Listing 11, I’ve given the same semantics that are used to typeset the paragraphs in this document, which are intended to be much more handout-y and much less essay-y. Try typesetting Listing 11 yourself and see what happens.

Lastly, one thing you might notice is that I sometimes put sentences on their own line.  $\LaTeX$  treats these sentences as being in the same paragraph because there is no blank line or `\par` between them. This isn’t strictly necessary; you’re more than welcome to put all of the sentences in a paragraph on one line, like I did in paragraph ‘five’.

However, there are two main reasons why it might be nice to put each sentence on its own line. First, having shorter lines might be easier to read, depending on how your editor is set up. Second, if you keep your `.tex` file under version control, it makes for cleaner diffs.<sup>21</sup>

---

<sup>21</sup>Explaining what exactly a version control system (VCS) is, is quite beyond the scope of this workshop. Basically, it’s a way to keep a history of all of the changes that have been made to a document. If you delve any further into learning  $\LaTeX$  or learning to program, you will probably also want to learn a VCS at some point. The most popular one is `git`. Also, hopefully you now understand this [joke](#).

```

\documentclass{article}
\setlength{\parindent}{0em}
\setlength{\parskip}{1ex}
\begin{document}

This is the first paragraph.
Wasn't that a great topic sentence?

Next paragraph please.
Paragraph number two is the best.
\par
The third paragraph will rule them all.
Sorry about the Lord of the Rings reference.

This concludes my five paragraph essay. As you can see, my conclusion definitely follows.

Yes I can count.
Jeeze.

\end{document}

```

Listing 11: Good practice for typesetting paragraphs

## 4 Useful stuff for linguists

### 4.1 fontspec and Unicode

In fn. 14, I mentioned a better way of typesetting diacritics in  $\text{\LaTeX}$ . This is where the `fontspec` package comes in. Historically, using  $\text{\LaTeX}$  with fonts hasn't really been a thing. This has changed recently with the advent of two other engines— $\text{\XeTeX}$  and  $\text{\LuaTeX}$ —and the `fontspec` package.

If you process a document with one of these two engines, you can use the `fontspec` package to specify which font you want to use. You can use any font that is installed on your computer.

In §6 of *A Beginner's Guide to  $\text{\LaTeX}$  (on the Mac)*, Alan gives the example of declaring a new font family to use for phonetic fonts. A complete example of what Alan suggests is given in Listing 12.<sup>22</sup>

```

\documentclass{article}
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{Times New Roman}
\newfontfamily\phonetic[] {Doulos SIL}
\usepackage{textglos} % this package provides semantic markup for inline linguistic examples

\begin{document}

The English word \xv{cat} is underlyingly {\phonetic/kæt/}.

\end{document}

```

Listing 12: Example of using a distinct font for phonetics

<sup>22</sup>See also [this question and its answers on TeX.SX](#). Note that in order for the example in Listing 12 to actually compile, you will need to have the font Doulos SIL installed on your machine. Covering how to install a font on your machine is beyond the scope of this workshop, since it is very specific to the type of operating system that you are running. However, you should be able to search for instructions online and pretty easily figure out how to do it.

In addition to using a separate font for phonetic stuff, you can also use one font for the entire document if the font you are using has glyphs for all of the characters that you need. I actually really like the Computer Modern font that is the default font in T<sub>E</sub>X. There is a version of the Computer Modern font that you can [download](#) and install on your machine which has glyphs for a huge range of the Unicode characters.

If you download and install this font on your machine, then you can do something like in Listing 13, rather than having to use a separate font for special (phonetic) characters.

```
\documentclass{article}
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{CMU Serif Roman}
\usepackage{textglos}
\begin{document}

The English word \xv{cat} is underlyingly /kæt/.
Also, look at the cool stuff that I can do in the same font: ášçëû!

\end{document}
```

Listing 13: Using one font that has a lot of Unicode glyphs

Processing either Listing 12 or 13 with either Lua<sub>La</sub>T<sub>E</sub>X or Xe<sub>La</sub>T<sub>E</sub>X will produce a PDF with the correct glyphs (as long as you have the requisite fonts installed on your computer). One further thing that is important for this to work correctly is to make sure that your .tex file has the correct character encoding. It is best to make sure that all of your .tex files are saved with UTF-8 encoding. A good editor should allow to see and change the character encoding of the file. Since this depends on the editor, it is beyond the scope of the workshop to explain it in any more detail, but you should be able to search online and figure it out.

## 4.2 Examples

There are two main packages that I would recommend for typesetting linguistic examples, **gb4e** and **ExPex**.<sup>23</sup> The **gb4e** package works well in most cases. For more complicated use cases, you might want to learn **ExPex**. However, this workshop will only focus on **gb4e**.

### 4.2.1 Basic linguistic example

Listing 14 gives an example of how to typeset some basic examples.<sup>24</sup> Try typesetting these examples yourself and see what the result is. Notice in particular that you can give the examples `\label`s and `\refer` to them inline just like with captioned things.

### 4.2.2 Glossing examples

With **gb4e**, you can also gloss examples. An example is shown in Listing 15.<sup>25</sup>

Try typesetting (1) yourself, giving it a `\label`, and `\referring` to it in the text of your document.

<sup>23</sup>There is also the package called **linguex**. I don't know much about it. I've always avoided it because, as Alan points out in *A Beginner's Guide to L<sup>A</sup>T<sub>E</sub>X (on the Mac)*, its markup is not all that semantic.

<sup>24</sup>Note that it is not strictly necessary to put each example in its own **exe** environment, but it is a good practice to do so, for at least two reasons. First, the markup is more semantic because **exe** is singular; it refers to *an* example, not a series of examples. Second, and more practically, it makes moving examples around a lot easier, either within the document or from one document to another.

<sup>25</sup>I'm a bit loathe to recommend this because the package currently has a bug, but it's a really great package. For an even better way of typesetting common linguistic gloss abbreviations than what you see in Listing 15, check out the **leipzig** package. If you use **leipzig** *without* the **glossaries** package, you shouldn't run into any trouble. However, if you use **leipzig** in conjunction with the **glossaries** package—which is a really great thing to do because it can then automatically generate a list of all glosses that you've used in your document—you will run into problems. There is a really hacky workaround [here](#), but it's a bug that should ultimately be fixed in the **leipzig** package. I've tried contacting the maintainer of the package, but I haven't gotten a response. I plan to try contacting her again soon, so hopefully the bug will be fixed at some point.

```

\documentclass{article}

\usepackage{textglos}

% \usepackage{fixltx2e} % only needed if you have TeX Live < 2015
\newcommand*{\Ind}[1]{\textsubscript{#1}}

\usepackage{gb4e}
\noautomath % you should always declare this after loading gb4e

\begin{document}
(\ref{ex:questionable-English}) is marginally acceptable.

\begin{exe}
  \ex[?]{His\Ind{i} mother loves every\Ind{i} boy no matter what.}
  \label{ex:questionable-English}
\end{exe}
\begin{exe}
  \ex[] {Strong crossover
    \begin{xlist}
      \ex[*]{He\Ind{i} loves everyone\Ind{i}}
      \ex[*]{She\Ind{i} thinks everyone\Ind{i} is smart}
      \label{ex:everyone-is-smart}
    \end{xlist}
  }
  \label{ex:strong-crossover}
\end{exe}

The examples in (\ref{ex:strong-crossover}) exemplify the phenomenon of strong crossover.
For example, in (\ref{ex:everyone-is-smart}), \xv{she} c-commands \xv{everyone}.
However, pronouns cannot c-command their binders.

\end{document}

```

Listing 14: Typesetting basic linguistics examples

```

\documentclass{article}
\newcommand*{\Fts}[1]{\textsc{#1}}
\usepackage{gb4e}
\noautomath
\begin{document}
\begin{exe}
  \ex[] {\gll Der Apfel würde gegessen.\\
    The.\Fts{m}.\Fts{sg}.\Fts{nom} apple was eaten\\
    \trans `The apple was eaten'
  }
\end{exe}
\end{document}

```

Listing 15: A glossed example with gb4e

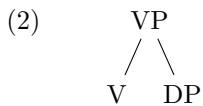
- (1) Hasan geu-peu-reubah aneuk miet nyan  
 Hasan 3POL-CAUS-fall child small DEM  
 ‘Hasan caused the child to fall’

### 4.3 Typesetting trees

There are two main packages that are useful for typesetting linguistics trees, `tikz-qtree` and `forest`. Both are built on top of the package, `tikz`. I would recommend using `forest`.<sup>26</sup> The syntax for both of these is almost the same, and `forest` is a lot more powerful. However, you don’t need to know its internals to do the basic stuff.

Let’s start with a very basic example, given in Listing 16, which will produce what you see in (2).

Try typesetting this yourself.



```

\documentclass{article}
\usepackage{forest}
\begin{document}

\begin{forest}
[VP
  [V]
  [DP]
]
\end{forest}

\end{document}
  
```

Listing 16: A very basic example with `forest`

There are a few things to note about this basic example.

First, line breaks are not necessary. You could have produced the same output by writing `[VP [V] [DP] ]`. Nonetheless, spacing things across lines is generally a good practice because it makes your code much more readable, which also in turn makes it much easier to debug if you are getting errors when trying to compile your file.

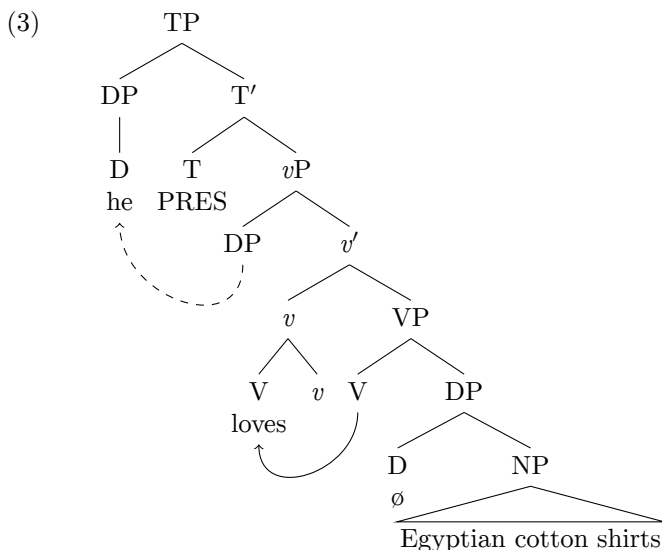
Second, you will see that this tree does not occur inside of a numbered example. This is something you will almost always want to do, as it is a standard in the field. Given what you learned in §4.2, you should be able to imagine how to do this. And we will see an explicit example of how to do this below.

Third, you will notice that the branches do not actually connect at the bottom of the “VP” node. Having the branches connect at the bottom of their parent node is the standard style in syntax. We will see how to draw a tree with this style below as well.

With that in mind, let’s take a look at a slightly more complicated example, given in Listing 17, which will address some of these issues. The code in Listing 17 will produce what you see in (3).

<sup>26</sup>For further discussion, see [this question and its answers on TeX.SX](#).





A few comments about what you see in Listing 17 are in order.

First, `forest` is designed to typeset trees as compactly as possible. Sometimes when you want to show movement, you will therefore need to increase the distance between siblings so that arrows don't overlap with something else. This is what `s sep+=` allows you to do. You can manually set the sibling separation with just `s sep=`, whereas `s sep+=` allows you to increase the default sibling separation value by a certain amount.

If you want to draw an arrow from one leaf in a tree to another leaf, you need to give them names that you can pass to the `\draw` command.

If you want to have spaces in a node in a tree, you will need to surround the content in braces, like what I did with `{Egyptian cotton shirts}` in Listing 17.

Finally, you will most likely want all of your trees to be typeset using both `baseline` and `qtree`. By declaring `\forestset{.style={qtree,baseline}}` in your preamble, you tell `forest` to typeset all of your trees like this.

The first option, `baseline`, ensures that the baseline of the typeset object is the top of the tree, rather than the bottom of the tree. This means that “TP” will be typeset as aligned with the example number (if you put it inside of an example environment) rather than “Egyptian cotton shirts”.


The second option, `qtree`, is a style that mimics the `qtree` package style for trees.<sup>27</sup> This is the style that is standard for typesetting trees. Specifically, it ensures that the branches of the tree connect at the bottom of each node. Try typesetting a tree without using the `qtree` option to see what happens.

## 4.4 Typesetting OT tableaux

There are two options that I would recommend for typesetting OT tableaux. One package is called `OTtblx`, which is in beta and not yet on CTAN. Thus, in order to use it, you need to put the `.sty` file in either the same directory as your `.tex` file or put it in a local folder that your  $\text{\TeX}$  distribution can see (cf. §2.3).<sup>28</sup>

To avoid this complication for the sake of an introductory workshop, we will instead use the package called `ot-tableau`, which is available on CTAN (*i.e.*, it is part of any good  $\text{\TeX}$  distribution, such as TeX Live or MiK $\text{\TeX}$ ). An example of how to typeset a tableau is given in Listing 18. This code will produce the result in (4).<sup>29</sup>

(4)

/bad/	*VOICED-CODA	IDENT-IO(voice)	*[+voi,-son]
a. bad	*!		**
 b. bat		*	*
c. pat		**!	
d. pad	*!	*	*

<sup>27</sup>Note that this style is not defined by `forest`. We must define it ourselves.

<sup>28</sup>Note that if you do decide to try `OTtblx` at some point, it must first be compiled to DVI format and then PDF format because it relies on the package `pstricks`. If you compile with  $\text{\XeLaTeX}$ , things should work fine, but it will not work with  $\text{\pdfLaTeX}$ .

<sup>29</sup>Disclaimer: I'm not a phonologist.

```

\documentclass{article}
\usepackage{forest}
\forestset{
  qtree/.style={ % define a style that imitates the qtree package
    for tree={
      parent anchor=south,
      child anchor=north,
      align=center,
      inner sep=1pt
    }
  },
  .style={ % declare styles to apply to all forest environments
    qtree,
    baseline
  }
}
\usepackage{gb4e}
\noautomath
\begin{document}
\begin{exe}
\ex{}{
  \begin{forest}
    [TP
      [DP
        [D\\he, name=DP2]
      ]
      [T$'$
        [T\\PRES]
        [\emph{v}P, s sep+=20pt
          [DP, name=DP1
            [\emph{v}$'$
              [\emph{v}
                [V\\loves, name=V2]
                [\emph{v}]
              ]
            [VP, s sep+=20pt
              [V, name=V1]
              [DP, s sep+=30pt
                [D\\ø]
                [NP
                  [{Egyptian cotton shirts}, triangle]
                ]
              ]
            ]
          ]
        ]
      ]
    ]
  ]
  \draw[->] (V1) [in=-90, out=-90, looseness=1.5] to (V2);
  \draw[->,dashed] (DP1) [in=-90, out=-90, looseness=1.5] to (DP2);
  \end{forest}
}
\end{exe}
\end{document}

```

Listing 17: An example of typesetting a syntax tree using `forest`

```

\documentclass{article}
\usepackage[
    shadedcells,
    notipa
]{ot-tableau}
\usepackage{gb4e}
\noautomath

\newcommand*{\Constr}[1]{\textsc{#1}}

\begin{document}

\begin{exe}
\ex[] {
\begin{tableau}{c|c|c}
\inp{/bad/}          \const{*Voiced-Coda} \const*{\Constr{Ident-IO}(voice)} \const*{*[+voi,--son]}
\cand{bad}           \vio{*!}              \vio{}              \vio{**}
\cand[\Optimal]{bat} \vio{}              \vio{*}              \vio{*}
\cand{pat}           \vio{}              \vio{**!}           \vio{}
\cand{pad}           \vio{*!}            \vio{*}              \vio{*}
\end{tableau}
}
\end{exe}

\end{document}

```

Listing 18: An example of an OT tableau

## 5 Getting help

As already mentioned, there is [TeX.SX](#). Don't forget to try searching the site before you ask your question. Chances are that somebody has already asked it. But we are generally pretty friendly and nice on TeX.SX, so don't hesitate to ask if you can't find an answer!

If you do decide ask a question on TeX.SX, in most cases you should provide a [Minimal \(non-\)Working Example \(MWE\)](#). The process of creating an MWE is often a good way to debug any problems you run into, and, in many cases, you might end up fixing the problem yourself in the course of creating an MWE.

There's also the [L<sup>A</sup>T<sub>E</sub>X Wikibook](#), which is generally pretty good.

If you're struggling with a particular package, try reading the documentation. Unlike most open-source software projects, L<sup>A</sup>T<sub>E</sub>X packages generally have *really good* documentation. You can find package documentation in a few places.

First, it's always on [CTAN](#).

CTAN isn't always the easiest to navigate, so two folks—Stefan Kottwitz and Paulo Cereda—have set up [TeXDoc Online](#), which allows you to easily search for package documentation by the name of the package.

TeXDoc Online is effectively just an online version of the command line tool, `texdoc`, which is part of TeX Live. If your distribution is TeX Live, you can just open a terminal and type `texdoc <package name>`.

And, of course, there are also some good books for learning L<sup>A</sup>T<sub>E</sub>X. There's a free one called the [The Not So Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>](#).<sup>30</sup> There's also [The L<sup>A</sup>T<sub>E</sub>X Companion](#).

<sup>30</sup>If you have TeX Live installed, you can find this book on your system by typing `texdoc lshort` at a terminal.