LaTeX workshop (for linguists)*

Adam Liter
tex@adamliter.org

Last updated: August 20, 2017

Contents

1 Terminology **1**

2 Setting up your machine **1**
2.1 Installing a TeX distribution . . . 1
2.1.1 Mac . . . 2
2.1.2 Linux . . . 2
2.1.3 Windows . . . 2
2.2 Keeping your TeX distribution up to date . . . 2
2.3 Local files . . . 3
2.3.1 TeX Live . . . 4
2.3.1.1 Create a `texmf` folder at TEXMFHOME . . . 4
2.3.1.2 Symlink your local `texmf` folder into TEXMFHOME . . . 5
2.3.2 MiKTeX . . . 5
2.3.3 An example use case . . . 6

3 General LaTeX stuff **7**
3.1 LaTeX philosophy . . . 7
3.2 Titles . . . 7
3.3 Quotes and dashes . . . 7
3.4 Formatting text . . . 9
3.5 Footnotes . . . 9
3.6 Special characters . . . 9
3.7 Math mode . . . 10
3.8 Tables . . . 10
3.9 Images . . . 11
3.10 Captioning and numbering . . . 11
3.10.1 Floats . . . 12
3.10.2 Non-float options . . . 13
3.11 Cross referencing . . . 14
3.12 Those annoying files . . . 14
3.13 Bibliographies . . . 15
3.13.1 Overview . . . 15
3.13.1.1 A terminological note . . . 15

*This handout was originally used as the basis for an impromptu LaTeX workshop that I gave at the 2015 Chicago Linguistic Institute. Since then, I have updated the `.pdf`, and I will continue to periodically keep it up to date. If you use this handout, I encourage you to occasionally check for a more up-to-date version of it because things do change. The most recent version of this `.pdf` can be found at https://bit.ly/latex-workshop. Furthermore, the `.tex` file that produced this `.pdf` is available at https://github.com/adamliter/latex-workshop. There are a few dependencies and oddities for compiling the document, which are explained in a README file found at that link. You're welcome to look at the source code of the document to get an idea of how to write something in LaTeX. Pull requests and/or suggested changes to the document are definitely welcome, too!

3.13.2 The `.bib` file . . . 16
3.13.2.1 Entry types . . . 16
3.13.2.2 Data fields . . . 17
3.13.3 The citation package . . . 19
3.13.3.1 `natbib` . . . 19
3.13.3.2 `biblatex` . . . 20
3.13.4 The backend processor . . . 24
3.13.4.1 BibTeX . . . 24
3.13.4.2 Biber . . . 24
3.14 Paragraphs . . . 24

4 Useful stuff for linguists **25**
4.1 `fontspec` and Unicode . . . 25
4.2 Examples . . . 26
4.2.1 Basic linguistic example . . . 27
4.2.2 Glossing examples . . . 27
4.3 Typesetting trees . . . 29
4.4 Typesetting OT tableaux . . . 32

5 Things to learn on your own **32**

6 Getting help **33**

7 Acknowledgments **34**

8 Feedback **34**

List of Listings

1 Make a minimal TDS-compliant directory at TEXMFHOME on OSX or Linux . . . 4
2 Make a minimal TDS-compliant directory at TEXMFHOME on Windows . . . 5
3 Make a TDS-compliant directory in Dropbox and symlink it into TEXMFHOME on OSX or Linux . . . 5
4 Make a TDS-compliant directory in Dropbox and symlink it into TEXMFHOME on Windows . . . 6
5 Make a TDS-compliant directory in Dropbox for MiKTeX on Windows . . . 6
6 Example of semantic markup in LaTeX for section headings . . . 8
7 A basic example of how to typeset the title of a paper using the `article` class . . . 8
8 A basic table in LaTeX . . . 10
9 A table in LaTeX using the package `booktabs` . . . 11
10 An example of including an image in a document . . . 11
11 Examples of floats in LaTeX . . . 12
12 Examples of tables and images as non-floats . . . 13
13 An example of referencing a figure in LaTeX . . . 14
14 Complete minimal working example showing how to use `natbib` and BibTeX . . . 21
15 Setting up citation and bibliography styles with `biblatex` . . . 22
16 A complete minimal working example with `biblatex` . . . 23
17 Really bad practice for separating paragraphs . . . 24
18 Good practice for typesetting paragraphs . . . 25
19 Example of using a distinct font for phonetics . . . 26
20 Using one font that has a lot of Unicode glyphs . . . 26
21 Typesetting basic linguistics examples . . . 27
22 A glossed example with `gb4e` . . . 28
23 An example with `leipzig` and `glossaries` . . . 28
24 A very basic example with `forest` . . . 30
25 An example of typesetting a syntax tree using `forest` . . . 31
26 An example of an OT tableau using `ot-tableau` . . . 33

1 Terminology

This section is largely a brief recap of §1 of Alan Munn's *A Beginner's Guide to LaTeX (on the Mac)*.[1] 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.

TeX 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.[2] In the computer world, it is common to shorten "distribution" to "distro"—*i.e.*, TeX distro.

Engines There are a few different engines that are standardly used to process a `.tex` file and turn it into a PDF, including pdfLaTeX, XeLaTeX, and LuaLaTeX.

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.

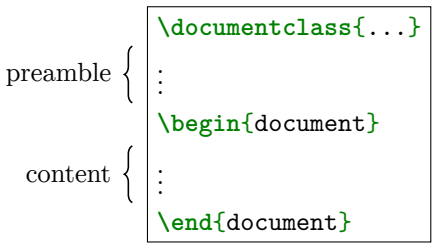


Figure 1: Schematic structure of a LaTeX document

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 TeX distribution

There are two relatively new and popular web editors for LaTeX—namely, ShareLaTeX and Overleaf. The web editors are useful tools for collaboratively working on a LaTeX document. They are also useful because you do not have to bother with installing your own TeX distribution on your computer.

Nonetheless, there are several advantages to installing a TeX distribution on your computer and being able to edit and compile `.tex` documents locally. The biggest advantage is being able to maintain a single master `.bib` file and use it in all of your `.tex` documents for references. For more on this, please read §2.3 and §3.13.

[1] 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.

[2] 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.

Another advantage is the ability to keep your TeX distribution up to date (see §2.2). Since the TeX distros used by these online editors often lag behind the update cycle of the various TeX distributions, you will be using packages that are a year or two old. If a bug in a package that you use is fixed by the package author but the online editor does not update its TeX distribution for a while, you won't get immediate access to that bug fix.

2.1.1 Mac

If you're on a Mac, you should install MacTeX.[3] MacTeX is all TeX Live underneath with just a thin wrapper that makes things work smoothly on a Mac. MacTeX 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 TeX 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

On Windows, you have two options: MiKTeX or TeX Live. For some time, the easiest thing to install on Windows was MiKTeX, which is a completely different distribution than TeX Live. This is because TeX Live was primarily designed for machines with an underlying Unix-based operating system.

Despite MiKTeX being easier to install there were, at one point, security concerns about MiKTeX and thus it was preferable to install TeX Live, even though it was much more complicated to install TeX Live on Windows.

Nowadays, these security concerns seem to have been mitigated,[4] and it's also now equally easy to install TeX Live on Windows. Therefore, from these two points of view, there isn't much reason to prefer one distribution over the other.

Most Windows users that I've encountered seem to have MiKTeX, probably because it was at one point more straightforward to install.

To install MiKTeX, go to the MiKTeX download page. If you download the "Basic Installer" from the download page, MiKTeX will just install the basic packages. I'd recommend instead downloading the "Net Installer" in order to install all of the packages from the beginning. This will greatly increase the time it takes to install, but you won't then have to go download a package from the internet every time you need to use a new package, which is what MiKTeX will have to do if you only do the basic installation.

If you'd rather install TeX Live for Windows, follow this link to download the `.exe` file to start the installation process. Note that TeX Live installs all of the packages from the beginning by default, so the installation process will probably take a few hours.

2.2 Keeping your TeX distribution up to date

It is good practice to periodically update your TeX distribution. A TeX distribution includes a bunch of packages, which are periodically edited by their maintainers. These packages are hosted on the Comprehensive TeX Archive Network (CTAN). You should thus periodically update things in case the maintainers of packages find a bug and fix

[3] If you're just installing MacTeX in 2017 on a new Mac, you should have nothing to worry about. However, if you had installed a TeX distribution on a Mac with Max OS X 10.10 (Yosemite) or earlier, you will want to read Herbert Schulz's document *MacTeX-2015 and El Capitan* before upgrading to a more recent version of macOS (10.11 or higher). If you do not follow the instructions in that document after you upgrade to El Capitan (10.11) or higher, you will run into problems.

[4] See this question and its answers on TeX.SX for discussion.

that bug or in case they add new features to the package.[5] To update your TEX distribution, refer to the question How do I update my TeX distribution? on TeX.SX.

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 TEX distribution.

For TeX Live, there is a new distribution that is released every year. The current one is TeX Live 2017.

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 saved 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 TEX Directory Structure (TDS) hierarchy. A minimal example of a directory structure that conforms to this standard is given in Figure 2.[6]

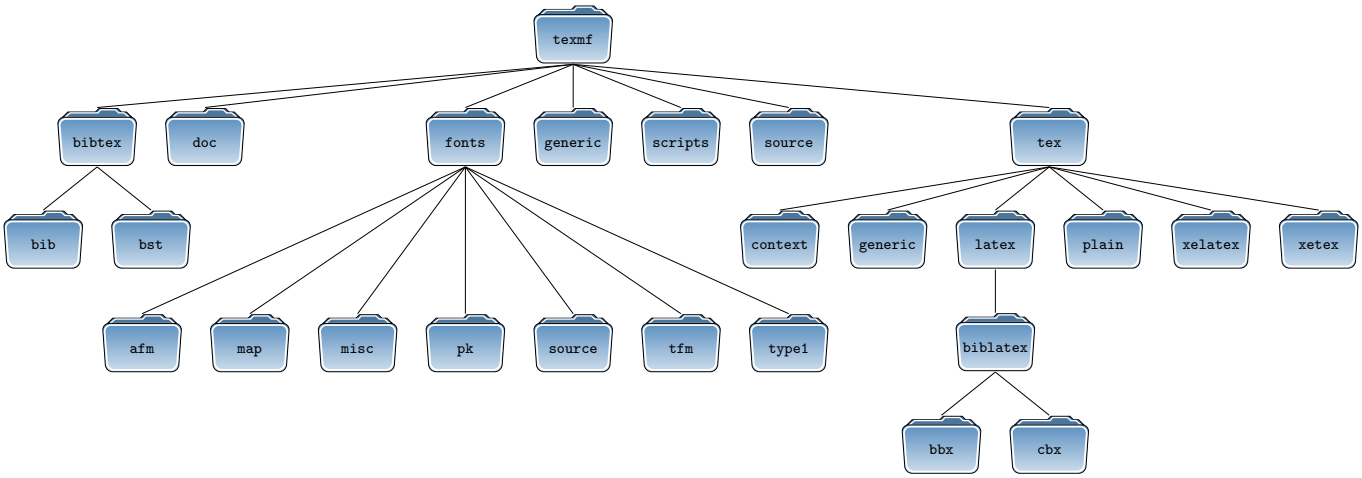


Figure 2: A minimal directory that conforms to the TDS standard

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.

Note that you should *only* put stuff in this directory that you want to be available to all of your `.tex` files, such as a master bibliography file, a custom package or style file that is not part of CTAN, *etc.*. This directory is *not* for your individual `.tex` documents. If you wish to learn more, see fn. 6.

[5] One thing that is also great about LATEX, 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 LATEX, you could typeset a LATEX file written in 1739 and the output you get would be identical to the output you got in 1739.

[6] 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.

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.

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.[7]

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.[8]

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 directory path: `/Users/<user name>/Library/texmf`.[9]

Even though the variable TEXMFHOME has a value, the folder might not exist. You need to create it. There are two options for doing this. You can create the folder in that exact location, or you can create the folder in a different location and make a symbolic link (symlink) at the location of TEXMFHOME that points to where the folder is actually located on your computer.

I would highly recommend the second option. If you do the second option, you could keep your `texmf` folder in the cloud with Dropbox, for example, which would allow you to have a backup of the folder as well as allow you to sync the folder across multiple machines.[10] §2.3.1.1 explains the first option, and §2.3.1.2 explains the second option.

2.3.1.1 Create a `texmf` folder at TEXMFHOME

Note that this method is discouraged. Instead, I suggest creating the folder somewhere else and making a symlink to it at TEXMFHOME. See §2.3.1.2 for instructions on how to do this.

OSX/Linux If you're on a Mac or Linux, you can open a terminal[11] and copy and paste the commands that are shown in Listing 1 and hit ENTER.[12]

```
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} && \
mkdir -p $(kpsewhich --var-value=TEXMFHOME)/tex/latex/biblatex/{bbx,cbx}
```

Listing 1: Make a minimal TDS-compliant directory at TEXMFHOME on OSX or Linux

[7] For discussion, see http://www.tex.ac.uk/FAQ-what-TDS.html.

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

[9] 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.

[10] Actually, what I would really recommend is keeping your local `texmf` folder under version control using, for example, `git`, and keeping it on GitHub or Bitbucket. However, explaining how to use a version control system is beyond the scope of this document. If you know how to use one, I assume you can extrapolate from the setup instructions given in §2.3.1.2.

[11] To open a terminal on a Mac, search Spotlight for the program "Terminal". It can also be found inside the Utilities folder, which is inside of the Applications folder.

[12] Note that the commands given in Listings 1–4 will only work after TeX Live has been installed, because the command line tool `kpsewhich` is part of TeX Live.

Windows If you're on Windows, you can copy and paste the commands shown in Listing 2 into the Command Prompt and hit ENTER.[13]

```
for /f "delims=" %I in ('kpsewhich --var-value=TEXMFHOME') do (
for %d in (doc, generic, scripts, source) do (
md "%I/%d")
for %d in (bib, bst) do (
md "%I/bibtex/%d")
for %d in (afm, map, misc, pk, source, tfm, type1) do (
md "%I/fonts/%d")
for %d in (context, generic, latex, plain, xelatex, xetex) do (
md "%I/tex/%d")
for %d in (bbx, cbx) do (
md "%I/tex/latex/biblatex/%d"))
```

Listing 2: Make a minimal TDS-compliant directory at TEXMFHOME on Windows

2.3.1.2 Symlink your local `texmf` folder into TEXMFHOME

An alternative to creating the `texmf` folder in the precise location that the variable TEXMFHOME points to is to instead create the folder in an alternative location, and then create a symlink at the value of TEXMFHOME that points to the `texmf` folder.

I would recommend this method because it allows you to create the `texmf` folder inside of your Dropbox folder, for example, which makes a backup of the `texmf` folder in the cloud and also allows you to sync your `texmf` folder across multiple machines.

In what follows, I give instructions for how to do this with Dropbox. If you wish to make the folder somewhere other than inside of your Dropbox folder, just replace the relevant bits of the directory paths in the commands that are given below.

OSX/Linux If you're on a Mac or Linux, the default location of your Dropbox folder should be `~/Dropbox`. Therefore, you can open a terminal and copy and paste the commands shown in Listing 3 and hit ENTER.

```
mkdir -p ~/Dropbox/texmf && \
cd ~/Dropbox/texmf && \
mkdir -p {doc,generic,scripts,source} && \
mkdir -p bibtex/{bib,bst} && \
mkdir -p fonts/{afm,map,misc,pk,source,tfm,type1} && \
mkdir -p tex/{context,generic,latex,plain,xelatex,xetex} && \
mkdir -p tex/latex/biblatex/{bbx,cbx} && \
ln -s ~/Dropbox/texmf $(kpsewhich --var-value=TEXMFHOME)
```

Listing 3: Make a TDS-compliant directory in Dropbox and symlink it into TEXMFHOME on OSX or Linux

Windows If you're on Windows Vista or up, you can copy and paste the commands shown in Listing 4 into the Command Prompt and hit ENTER. If you're on Windows XP, why are you still running Windows XP? (Symbolic links don't work on Windows XP.)

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.

[13] You can open the Command Prompt on Windows by searching for it in the start menu.

```
md "%USERPROFILE%\Dropbox\texmf"
chdir "%USERPROFILE%\Dropbox\texmf"
for %d in (doc, generic, scripts, source) do (md "%d")
for %d in (bib, bst) do (md "bibtex\%d")
for %d in (afm, map, misc, pk, source, tfm, type1) do (md "fonts\%d")
for %d in (context, generic, latex, plain, xelatex, xetex) do (md "tex\%d")
for %d in (bbx, cbx) do (md "tex\latex\biblatex\%d")
for /f "delims=" %I in ('kpsewhich --var-value=TEXMFHOME') do (
mklink /J "%I" "%USERPROFILE%\Dropbox\texmf")
```

Listing 4: Make a TDS-compliant directory in Dropbox and symlink it into TEXMFHOME on Windows

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`.

However, since it doesn't really matter, I would instead recommend creating the `texmf` folder inside of a Dropbox folder so that you can have a backup of your `texmf` folder in the cloud and also sync it across machines if you have multiple machines.

To do this, you can copy and paste the commands shown in Listing 5 into the Windows Command Prompt and hit ENTER.[14]

```
md "%USERPROFILE%\Dropbox\texmf"
chdir "%USERPROFILE%\Dropbox\texmf"
for %d in (doc, generic, scripts, source) do (md "%d")
for %d in (bib, bst) do (md "bibtex\%d")
for %d in (afm, map, misc, pk, source, tfm, type1) do (md "fonts\%d")
for %d in (context, generic, latex, plain, xelatex, xetex) do (md "tex\%d")
for %d in (bbx, cbx) do (md "tex\latex\biblatex\%d")
```

Listing 5: Make a TDS-compliant directory in Dropbox for MiKTEX on Windows

After doing this, click Start → Programs → MiKTEX 2.9 → Maintenance → Settings to open the MiKTEX Options window. Do *not* open Settings (Admin), just open Settings.[15] Next, click on the Roots tab. Click Add in order to add a local `texmf` folder.

Navigate to `C:\Users\<user name>\Dropbox\texmf` and click OK.

Next, click Apply.

Then, click on the General tab. Click on Refresh FNDB. Then click OK.

See also here and here for instructions with screenshots.

2.3.3 An example use case

As mentioned above, this local `texmf` folder that you just created is only for local files that you want to be available for use in all of your `.tex` documents. For example, this `texmf` folder is where you would put local style or class files that are not part of your TEX distribution.[16]

Perhaps the most important use case for the local `texmf` folder is for maintaining a single master bibliography file that can be used for citations in all of your documents. This will be discussed in detail in §3.13.

For the moment, however, I will briefly discuss a use case that is relevant to linguists who use LATEX.

[14] While MiKTEX recommends using the folder name `Local TeX Files`, it really doesn't matter what folder name is used. The commands in Listing 5 will use the folder name `texmf` instead of `Local TeX Files`. This way, if you also happen to have a Mac or Linux machine, you can use the same folder inside your Dropbox folder on that other computer, too.

[15] For a discussion of the differences between administrative and user mode in MiKTEX, see this question and its answers on TeX.SX.

[16] To reiterate, do *not* put any of your `.tex` documents inside this `texmf` folder.

Alexis Dimitriadis has modified part of a package called `gb4e`, which you will be introduced to in §4.2. `gb4e` is one of the recommended packages for typesetting linguistic examples.

One drawback about the `gb4e` package is that it does not provide a straightforward way to indicate which language an example comes from, at least not out of the box. The modifications that Alexis Dimitriadis has made allow you to do this straightforwardly. He has named the modified package `cgloss`.

However, `cgloss` is not available as part of TeX Live or MiKTEX. It is only available on Alexis's website.

If you want to use Alexis's modifications, you will need to download the file `cgloss.sty` and place it in your local `texmf` folder. Specifically, it should be placed inside of the folder `texmf/tex/latex`. The reference to `texmf` in the preceding sentence should be treated as a placeholder for wherever you created your local `texmf` folder. So, for example, if you created your local `texmf` folder in Dropbox on OSX or Linux, then you should put the file inside of `~/Dropbox/texmf/tex/latex`.

This will now allow you to use the `cgloss` package in your `.tex` documents, even though the package is not part of your TEX distribution.[17] In other words, you can put any custom packages or style files into the local `texmf` directory.

3 General LATEX stuff

3.1 LATEX philosophy

This subsection is very similar to §3 of Alan Munn's *A Beginner's Guide to LATEX (on the Mac)*.

LATEX 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 LATEX 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 LATEX 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 individual section heading.[18] 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 6.

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 7.

3.3 Quotes and dashes

One idiosyncrasy of 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 `.

[17] After you have read §4.2 and learned how to use the `gb4e` package, you can read Alan Munn's answer on TeX.SX about how to use the functionality of `cgloss`.

[18] 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 LATEX, you should at least try to get them to use semantic markup in Word if they don't already do so. :)

```latex
\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 6: Example of semantic markup in LaTeX for section headings

```latex
\documentclass{article}

\title{Super awesome title}
\author{Best Author}
\date{July 22, 2015} % if you want today's date, replace July 22, 2015 with \today

\begin{document}

\maketitle

\section{Introduction}

Blah blah.

\end{document}
```

Listing 7: A basic example of how to typeset the title of a paper using the `article` class

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 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.[19]

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}`.[20]

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. This is called semantic markup, and it makes any changes you might want to make to the formatting of a document much easier.

3.5 Footnotes

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

3.6 Special characters

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

If you ever want to print any of these characters in the output, you need to escape them with '`\`'.[21]

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

[19] 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`.

[20] This is just intended as an example of semantic markup. I would not actually recommend typesetting glossed features in this manner; instead of defining your own `\Fts` macro, you should use the `leipzig` package. This package provides commands for many of the common feature abbreviations, all following the Leipzig Glossing Rules, which is a standard for glosses in the field of linguistics. In addition to providing glossing macros for all of the glosses in the Leipzig Glossing Rules, `leipzig` also provides a way to automagically print all of the glosses used in a document and their meanings, via the `glossaries` package. This package will be covered in more detail in §4.2.2.

[21] One exception to this is the escape character itself, '`\`', because the sequence '`\\`' has a special meaning in 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 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 6).

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 LaTeX 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.[22] For example, `$\lambda x$` will produce λx.

3.8 Tables

Tables are admittedly a bit of a pain in the ass in LaTeX. Typesetting them takes a while to get used to. Let's see an example of a basic table, such as the one in Listing 8.

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

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

\end{document}
```

Listing 8: A basic table in LaTeX

The code in Listing 8 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 9, which produces the following output.

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

[22] This is a bit of an overstatement. There are other use cases. Typesetting feature bundles in morphosyntax is one such use case, for example. However, for the purposes of an introductory workshop, you should really just know what math mode is.

```
\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 9: A table in LaTeX using the package `booktabs`

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.[23] Let's look at the example in Listing 10.

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

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

\end{document}
```

Listing 10: 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.[24] 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`.[25] 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.

[23] Notice that the name of the package is `graphicx` with an 'x', not `graphics`. The `graphics` package is different from `graphicx`, and the example in Listing 10 will not work if you use the `graphics` package by accident.

[24] 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.

[25] 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 automagically numbering tables and figures, floats also allow us to provide a caption. One thing to know about floats is that LaTeX has a special way of handling how they are typeset. Suffice it to say, it's rather complicated.[26] All you need to know is that 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 LaTeX's algorithm has not done a good job. However, while you're writing a document, leave all four options and let 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. LaTeX will automagically 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 the two examples in Listing 11. 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 11: Examples of floats in LaTeX

[26] 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 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 12. 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 \\ \midrule
        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 12: 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 13.

```
\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 13: 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.[27]

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.[28]

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 `\refer`enced 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

[27] If you really like automagic, you might want to check out the `cleveref` package.

[28] Spaces are technically allowed in the names of `\label`s, but they are highly discouraged. In certain cases, spaces in `\label` names might cause problems.

up. So, if you tried typesetting your document, received a compilation error, figured out what caused the error, and 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.

In order for this to work correctly, you will first need to set up a local `texmf` folder. How you do this depends on which TeX distribution you have. There are explicit instructions in §2.3. If you have not already set up a local `texmf` folder on your machine, go there and follow the instructions.

3.13.1 Overview

There are several components that are relevant to setting up your machine in order to be able to use a single master bibliography file (`.bib`) for all of your references in all of your `.tex` documents.

The `.bib` file You need to create a bibliographic database that contains all of the information for all of the references that you wish to cite in your `.tex` documents. This file is just a plain text file, but there are graphical user interfaces that can be used to edit the file, such as JabRef or BibDesk.[29] How to manage this file and other relevant aspects will be discussed in detail in §3.13.2.

The citation package Strictly speaking, you do not need to use a citation package. The base LaTeX format already provides the command `\cite{...}`. However, citation packages provide much more flexibility and many more features than what is offered by the base LaTeX format. Citation packages are also great for easily changing the style that your references are printed in. They also often provide more specific commands for inline citations and parenthetical citations. Some examples of citation packages include `cite`, `apacite`, `natbib`, and `biblatex`. For the purposes of linguistics at least, I would only recommend `natbib` or `biblatex`. Both of these packages will be discussed in more detail in §3.13.3.

The backend processor In addition to using a citation package, you will also need to use a program for getting the relevant citations from your `.bib` file. Using a `.bib` file for references involves a series of compilation steps. You must first compile your `.tex` file to produce either an `.aux` or `.bcf` file. Then you must process the `.aux` or `.bcf` file with a backend processor. The two backend processors on the market are BibTeX and Biber. Processing the `.aux` or `.bcf` goes through and matches up the things that were cited in your `.tex` file with entries in your `.bib` file. The differences between the two backend processors will be discussed in more detail in §3.13.4. The necessary compilation steps will actually be discussed in §3.13.3 because it depends on the citation package that you use, not the backend processor.

3.13.1.1 A terminological note

Before proceeding any further, it's worth touching briefly upon a bit of history to hopefully avoid some terminological confusion. However, what is perhaps more likely, is that this brief excursion won't make any sense. If that's the case, don't worry. That's not your fault. The history of TeX and the evolution of relevant terminology is complicated.

Hopefully the terminology won't cause too much confusion, but I would nonetheless like to make note of it. If it doesn't make sense now, perhaps it will in 6 months.

For a while, the only option for a backend processor was BibTeX. BibTeX was developed in the ancient days when memory and hard drive space were expensive, character encodings were limited, and dinosaurs still roamed the earth.

Biblatex is a recent and modern alternative to BibTeX. People sometimes use Biblatex to refer to the combination of the citation package `biblatex` and the backend processor Biber. I will adopt this practice of using Biblatex to refer to both of these things in combination, whereas `biblatex` will only be used to refer to the citation package.

[29] BibDesk is only available on a Mac. It is installed by default when you install MacTeX, so it is probably already on your computer.

What makes this practice particularly confusing, is that many people use Biblatex in contradistinction to BibTeX. The more accurate contrast, however, would be to contrast Biber with BibTeX.

For better or worse, however, this terminological parlance is quite common among TeX users and is probably here to stay. One thing that is actually useful about this seemingly confusing terminology, however, is that there are differences between how you write a `.bib` file that is intended to be used with Biblatex and how you write a `.bib` file that is intended to be used with BibTeX.

In other words, it can actually be useful to distinguish Biblatex and BibTeX if you want to know how a `.bib` file was prepared, for example. Biblatex supports a wider range of entry types and a wider range of data fields and is thus much more flexible and versatile than BibTeX.[30] Biblatex also plays much nicer with accented Latin characters and non-Latin alphabets than does BibTeX.

A full discussion of these differences is beyond the scope of this document. Some of the differences will be briefly discussed below, but for further discussion, the reader is referred to the following resources.

- The question bibtex vs. biber and biblatex vs. natbib and its answers on TeX.SX
- The question What to do to switch to biblatex? and its answers on TeX.SX
- The question Compatibility of bibtex and biblatex bibliography files? and its answers on TeX.SX
- §§1–3 of the `biblatex` documentation

3.13.2 The `.bib` file

In order to get started, you need to create a `.bib` file. This is just a plain text file, but your life will probably be made much easier if you use a graphical user interface to edit the file.

JabRef is a crossplatform option. Follow the link to download and install it.

If you have a Mac, you can use BibDesk. It should already be installed if you installed MacTeX.

Another thing that is nice about these graphical user interfaces is that they allow you to pair `.pdf` files to the entries in your database. If you pair a `.pdf` with an entry, then you can open JabRef or BibDesk and click on the link to the `.pdf` file in order to open it without having to find it on your computer.

Before creating a database, you will want to make sure that you save the `.bib` file with UTF-8 encoding. For some reason, the default character encoding in JabRef (on Windows at least) is Cp1252. This will just give you a lot of headaches. Before you create your `.bib` file, go into the JabRef preferences and change the default character encoding to UTF-8.

You should do the same with BibDesk. Before creating your `.bib` file, open BibDesk, go to preferences and set the default character encoding to UTF-8.

If you have trouble finding these options in either BibDesk or JabRef, Google it. `:p`

Once you have changed the default encoding, create a new bibliography file (`.bib`). It is *very important* that you save it in the correct location. It *must* go inside your local `texmf` folder; specifically, it *must* go inside `texmf/bibtex/bib`. If you do not have a local `texmf` folder set up, go read and follow the instructions in §2.3.

Moreover, you will also save yourself a lot of pain if you *do not use spaces* in the name of the file. I would recommend something simple like `master.bib` or `linguistics.bib`.

Now that you have your `.bib` file setup, you can start adding entries to it.

3.13.2.1 Entry types

There are various entry types, and the available entry types differ slightly depending on whether you use `biblatex` or not.[31] In general, you will probably mostly only need to use the entry types `@article`, `@book`, `@incollection`,

[30] You will be introduced to entry types in §3.13.2.1 and data fields in §3.13.2.2.

[31] Covering this in any detail is really beyond the scope of this document. For more information, please see the resources mentioned in §3.13.1.1. For the most part, however, you do not need to worry about it. A `.bib` file that was prepared for use with BibTeX will work just fine with Biblatex. You just will not be able to use some of the more flexible and versatile features of Biblatex (which you probably won't need anyway,

`@inproceedings`, `@misc`, and `@unpublished`.

`@article` This entry type should be used for journal articles.

`@book` This entry type should be used for complete books.

`@incollection` This entry type should be used for chapters in edited volumes. Do *not* use the `@inbook` entry type for this.

`@inproceedings` This entry type should be used for stuff that is published in conference proceedings.

`@misc` This entry type should be used for miscellaneous stuff, such as talks or posters presented at conferences (that aren't published in proceedings). You can add something to the `howpublished` field that indicates where it was presented and whether it was a talk or a poster. See §3.13.2.2 for more details.

`@unpublished` This entry type can be used for unpublished manuscripts.

3.13.2.2 Data fields

Inside of each entry, there are certain data fields that you need to fill out. The available data fields depend on which entry type you are using.[32] In general, the bare minimum that you need to fill in are the data fields `author`, `title`, and `year`.

For each of the entry types mentioned above, here are the recommended data fields that I would minimally suggest filling in.

- `@article`
  - `author`
  - `title`
  - `journal`
  - `year`
  - `volume`
  - `number`
  - `pages`
  - `doi`
- `@book`
  - `author`
  - `title`
  - `publisher`
  - `year`
  - `address`
- `@incollection`
  - `author`
  - `title`
  - `booktitle`
  - `publisher`
  - `year`
  - `editor`
  - `pages`
  - `address`
- `@inproceedings`
  - `author`
  - `title`
  - `booktitle`
  - `publisher`
  - `year`
  - `pages`
  - `address`
- `@misc`
  - `author`
  - `title`
  - `howpublished`
  - `year`
- `@unpublished`
  - `author`
  - `note`
  - `title`
  - `year`

Keywords Another useful data field is `keywords`. This can be used with every entry type, and I would highly recommend filling it in for every entry.

You can make up the keywords that you use. Doing so allows you to easily sort and search through your `.bib` file.

For example, you could put keywords in all of your entries for different subdisciplines, like syntax, or semantics, or phonology, *etc.*. If you wanted to look through your `.bib` file for a certain syntax paper that you cannot remember, for example, then you could just look at all of the entries that contain syntax in the `keywords` data field.

except for complicated use cases).

[32] The available data fields also depend on whether you are preparing your `.bib` file for use with Biblatex or BibTeX (cf. fn. 31). A full discussion of the many more data fields that Biblatex provides is much beyond the scope of this document. You largely do not need to worry about it unless you would like to use some of Biblatex's more advanced features. The data fields discussed here are for a BibTeX `.bib` file, but they will all work just fine if you use Biblatex instead of BibTeX.

On names The method for entering names into the `author` and `editor` data fields is a bit idiosyncratic, but it allows names to be typeset in a very fine-grained manner, so it is well worth it.

There are four components to a name.

(i) First name (this includes any and all middle names)

(ii) The "von" part (examples include "von", "van", "de", "de la", *etc.*)

(iii) The last name (does *not* include the "von" part)

(iv) The "Jr" part (examples include "Jr.", "Sr.", "IV", *etc.*)

There are three possible ways to enter names into a `.bib` database.

(i) "First von Last"

(ii) "von Last, First"

(iii) "von Last, Jr, First"

I would highly recommend always using either the second or third option. The first option does not work if an author has multiple *last* names and no "von" part in their name.

Finally, the last thing to know about names is that multiple authors or editors should be separated with `and`.

So, for example, if something you wish to cite has three authors, you should write the following in the `author` data field: `Matthewson, Lisa and von Fintel, Kai and Smith, Jr., Mary`.

On casing Depending on the style that you use for your bibliography, you might see your titles being typeset in sentence casing rather than title casing. Sentence casing is what's called for by the Unified Stylesheet for Linguistics Journals, for example.

Nonetheless, there might be certain things that ought to remain capitalized. In this case, you should surround these things with braces in the data field.

For example, if you are adding an `@incollection` entry type whose `title` is *On the absence of certain quantifiers in Mohawk*, you should enter it into the `title` field as `On the Absence of Certain Quantifiers in {Mohawk}`. Note that the braces should surround the entire word, *not* just the letter M. If the braces do not surround the entire word, it will mess up the kerning.

Note furthermore that the data field was written using title casing, but the output you see will be sentence casing, except for the word *Mohawk*, because we surrounded it with braces. I would recommend always entering titles into the data fields using title casing. Both `biblatex` and BibTeX provide means for automagically converting from title casing to sentence casing, depending on the style, but not necessarily vice versa. Thus, you will save yourself much pain if you need to switch to a style that uses title casing if the entries in your `.bib` file are already in title casing.

On cite keys In addition to filling out all of the data fields, you will also need to give each entry a unique cite key. This is what you will use to cite the entry in one of your `.tex` documents.

You can come up with your own algorithm for determining cite keys, but it is best to use some sort of algorithm rather than making up arbitrary cite keys as you go along. Here are two suggestions.

(i) `lastnameYYYY`

(ii) `lastnameYYYY:informativewordfromtitle`

The first one is very basic. You could just use the last name of the first author and the four-digit year of the publication for the cite key. You will of course run into trouble when an author has multiple publications in the same year. In this case, you could do something like `lastnameYYYYa` and `lastnameYYYYb` to disambiguate the two publications.

However, it might be preferable to do something like the second option instead. If you have to change the cite key after you've already added it and used it, then you would need to go back and change how you cited it in your `.tex` file. If you use the second suggested algorithm instead, it's very unlikely that you will run into a situation where you might have duplicate cite keys for distinct entries.

3.13.3 The citation package

The two main options for citation packages that are recommended are `natbib` and `biblatex`. There are others, but they are nowhere near as versatile as these two.

3.13.3.1 `natbib`

`natbib` is only compatible with using BibTeX as the backend processor for your `.aux` and `.bib` files (see §3.13.4). Since BibTeX was developed a long time ago, it does not play all that nicely with accented Latin characters and non-Latin alphabets.

I would highly recommend using `biblatex` and Biber instead of `natbib` and BibTeX. However, it is worth noting that academic journals that accept LaTeX submissions often require bibliographies that use `natbib` and BibTeX.[33]

Citation commands There are two main commands for citations with `natbib` that you ought to know about. You can use `\citet[][]{...}` for inline citations and `\citep[][]{...}` for parenthetical citations.

The first optional argument of both commands can be used to specify a prenote, and the second optional argument can be used to specify page numbers.

There is also a handy `natbib` reference sheet that describes more of the citation commands.

Styles `natbib` and BibTeX use `.bst` files to specify different bibliography styles. `.bst` files are awful.

Thankfully, Bridget Samuels has gone through the trouble of creating one for us that conforms to the Unified Stylesheet for Linguistics Journals. If you decide to use `natbib` and BibTeX, you should download the file `unified.bst` and place it in your local `texmf` directory. Specifically, it needs to go into the folder `texmf/bibtex/bst` (see §2.3 if you've somehow made it this far without setting up a local `texmf` folder).

There are many other styles that you can use out of the box with either a TeX Live/MacTeX or MiKTeX distribution, such as `alpha` or `apalike`. But, presumably, you will want to use the `unified` style that Bridget Samuels wrote since you're (presumably) a linguist.

Compilation steps In order for the references to be handled automagically in your document, you need to go through a series of compilation steps. This series of steps might depend on the citation package that you use and perhaps also the style.

In most cases when using `natbib`, it is necessary to do four compilation steps.

(i) `latex myfile.tex`

(ii) `bibtex myfile.aux`

(iii) `latex myfile.tex`

(iv) `latex myfile.tex`

In these four steps, you should treat `latex` as a placeholder for whatever engine you are using. For example, if you are using pdfLaTeX, then you should replace `latex` with `pdflatex` in these four steps. Similarly, if you are using XeLaTeX, then you should replace `latex` with `xelatex` in these four steps.

These commands can be run from the command line, but, more likely, you will just run these commands from inside of your editor.

Note that the file extensions are optional. In fact, it's probably better practice to omit them and let the program try to guess which file to process. I've only included them here to draw your attention to the fact that the `.aux` file is what you should process with BibTeX, *not* the `.tex` file or the `.bib` file.

[33] A notable exception to this is *Semantics & Pragmatics*, which accepts submissions that use `biblatex` and Biber.

Again, if you omit the `.aux` file extension, BibTeX will do things correctly, so just go ahead and omit it if you do do your build steps on the command line. More likely, you will just run these commands from within your editor, in which case you won't need to worry about the file extensions.

The first pass of `latex` will collect all of the cite keys that you are trying to cite inside of your `.tex` document and put them in the `.aux` file.

The pass of `bibtex` over the `.aux` file goes and gets the citations from your `.bib` file that match the cite keys in the `.aux` file.

The second pass of `latex` inserts the citations into the generated output, and the third pass of `latex` resolves any cross references that have changed.

While these series of compilation steps may seem a bit annoying, it is necessary for LaTeX to do its automagic.[34] You will definitely come to appreciate how LaTeX handles bibliographies and references compared to Microsoft Word.

Complete minimal working example Listing 14 gives a complete minimal working example of how to use `natbib` and BibTeX.

There are a few things to note about this example.

First, the bibliography file and the style file are specified at the end of the document, just before `\end{document}`.[35] In both cases, be sure to omit the file extensions (*i.e.*, do *not* add `.bib` or `.bst`). You can use multiple bibliography files, but you should only have one call to the `\bibliography` command. If you need to use multiple `.bib` files, you should pass a comma-separated list to the `\bibliography` command, such as `\bibliography{linguistics,philosophy}`. This command would allow you to use the two bibliography files `linguistics.bib` and `philosophy.bib`.

Second, the `.bib` file is bundled into this example so that you can copy and paste it into an editor and compile it directly. But this is *not* how you should write your `.tex` files. You should instead use a single master bibliography file as discussed in §3.13.2.

In this case, you would replace `\bibliography{\jobname}` with `\bibliography{master}`, if, for example, you named your `.bib` file `master.bib`. Moreover, you would also then completely delete everything from `\begin{filecontents}` to `\end{filecontents}`.

Try compiling this example using what you just learned above about how to compile a `.tex` document that uses `natbib` for citations.

3.13.3.2 `biblatex`

Unlike `natbib`, `biblatex` can be used with either backend processor, Biber or BibTeX. If you wish to use the full range of features that `biblatex` provides, you will need to use Biber.

As mentioned above, there is good reason to use `biblatex` and Biber over `natbib` and BibTeX. Most notably, `biblatex` and Biber work a lot more straightforwardly with accented Latin characters and non-Latin alphabets.

In addition to not being an acceptable format for submission to many academic journals, the only available implementation of the Unified Stylesheet for Linguistics Journals in `biblatex` is currently very early in development. However, at the time of the most recent update to this `.pdf`, the implementation is much better than it used to be.

Citation commands `biblatex` provides `\textcite[][]{...}` for inline citations and `\autocite[][]{...}` for non-inline citations. Like with the `natbib` citation commands, the first optional argument can be used for a prenote, and the second optional argument can be used to specify specific page numbers.

One thing that is nice about `biblatex` is that the `\autocite` command is a high-level command for non-inline citations. In the case of linguistics, at least, you will usually want this to be a parenthetical citation. However, some styles call for footnotes for non-inline citations instead of parenthetical citations. If you use `biblatex` and the `\autocite` command, all you have to do is change the style, and this will be handled automagically for you.

[34] See §5 for some suggestions on how to automate the compilation process.

[35] Technically, you can put the `\bibliographystyle` command anywhere in your document, but I like to keep it next to the `\bibliography` command, which is what prints the references section and thus must come at the end of the document.

```
\documentclass{article}

\begin{filecontents}{\jobname.bib}
@book{chomsky1995:MP,
    Address = {Cambridge, MA},
    Author = {Chomsky, Noam},
    Publisher = {The MIT Press},
    Title = {The Minimalist Program},
    Year = {1995}}
\end{filecontents}

\usepackage{natbib}

\begin{document}

In Minimalist syntax, S-Structure has been eliminated \citep[see][73--124]{chomsky1995:MP}.

\bibliography{\jobname}
% the following will only work if unified.bst is in your local texmf folder
% if you haven't downloaded that file yet, just replace unified with
% apalike and then try compiling this example
\bibliographystyle{unified}
\end{document}
```

Listing 14: Complete minimal working example showing how to use `natbib` and BibTeX

To learn more about these citation commands and others that `biblatex` provides, see §3.8 of the biblatex documentation.

Styles `biblatex` provides many predefined styles that control both the citation style and the bibliography style. The citation style is controlled by a `.cbx` file, and the bibliography style is controlled by a `.bbx` file.

The style options are declared globally when the `biblatex` package is loaded. Assuming that a style provides both a `.cbx` and a `.bbx` file with the same name, you can declare these at the same time by setting `style` to the name of the style that you wish to use.

If you wish to mix and match citation and bibliography styles, you can instead declare them separately, with `citestyle` and `bibstyle` respectively. For an example, see Listing 15.

As mentioned above, the one implementation of the Unified Stylesheet for Linguistics Journals for `biblatex` is currently very much in an alpha stage of testing. This implementation is maintained by *Semantics & Pragmatics*, and it is available here.

At the time of the most recent update to this `.pdf` (August 20, 2017), the implementation is much better than it used to be. If you would like to use `biblatex` (which I would recommend for the reasons already discussed), I would recommend using *S&P*'s implementation of the Unified Stylesheet for Linguistics Journals.

Nonetheless, if you don't want to download it and put it in a local `texmf` folder, then you can use the `authoryear-comp` citation style and the `authoryear` bibliography style that are both provided by `biblatex` out of the box. The code given in Listing 15, which shows how to set these styles when loading `biblatex`, will produce an approximation of the Unified Stylesheet for Linguistics Journals that is probably good enough for all purposes other than submitting to a journal.[36]

The code in Listing 15 only shows how to set up the style. For a complete minimal working example, see Listing 16.

[36] And, I mean, bibliography and citation styles are just prescriptivist nonsense anyway. As long as all of the information is there, the style doesn't call for underlining, and the formatting is consistent, who cares ... ?

```
\documentlcass{article}
\usepackage[
    backend=biber,% you can use bibtex instead of biber if you want
    citestyle=authoryear-comp,
    bibstyle=authoryear
]{biblatex}
% This is needed because the default authoryear style puts "In:"
% before a journal name, which is not standard in linguistics styles.
% So this code gets rid of that.
\renewbibmacro{in:}{%
  \ifentrytype{article}{}{\printtext{\bibstring{in}\intitlepunct}}}

\begin{document}

The preamble of this document shows how to approximate a linguistics style.

\end{document}
```

Listing 15: Setting up citation and bibliography styles with `biblatex`

Compilation steps Just like with `natbib`, `biblatex` requires a certain series of compilation steps in order for everything to work. Again, the steps that are necessary might depend on certain things, such as which style you use.[37]

In most cases when using `biblatex`, it is sufficient to do the following three compilation steps.

(i) `latex myfile.tex`

(ii) `biber myfile.bcf` OR `bibtex myfile.aux`[38]

(iii) `latex myfile.tex`

Just like we discussed above for `natbib`, you should treat `latex` as a placeholder for whatever engine you are using. So, for example, if you are using pdfLaTeX, then you should replace `latex` with `pdflatex` in these three steps. Similarly, if you are using XeLaTeX, then you should replace `latex` with `xelatex` in these three steps.

Likewise, these are commands that you could enter into a terminal, but in most cases you will just run them from within your editor.

Were you to run these commands in a terminal, a further similarity to what we discussed above for `natbib` is that you could leave off the file extensions. In fact, I would encourage you to leave off the file extensions if you do end up doing your build steps inside of a terminal.

I have only shown the file extensions here in order to draw your attention to the fact that Biber processes a `.bcf` file rather than the `.aux` file. Rather than having to remember, this, though, you could just leave off the file extension and let the program guess which file to process.

The first pass through `latex` is necessary to generate the `.bcf` file.[39] This serves a similar purpose to the `.aux` file in the case of BibTeX. Specifically, it allows Biber to get all of the citations from your `.bib` file that match the cite keys that you used in your `.tex` document.

The second pass through `latex` typesets all of the references.[40]

[37] For example, I think citation styles that do non-inline citations as footnotes might require a third pass of `latex`.

[38] As discussed above and as will be discussed further below in §3.13.4, it is possible to use either backend with `biblatex`. Because Biber has many advantages over BibTeX, I would always recommend using Biber. Nonetheless, if you choose to use BibTeX with `biblatex` for whatever reason, then you will need to run `bibtex` on the `.aux` file instead of running `biber` on the `.bcf` file.

[39] Or the `.aux` file if you're using BibTeX as the backend (cf. fn. 38).

[40] Unlike with `natbib`, you do not need a third pass through `latex` in most cases. This is because `biblatex` loads all of the references into the preamble of your document, and so it can determine the cross references before processing the document. In other words, you do not need a third pass through `latex` (in most cases) to sort out any cross references that have changed, like you do with `natbib`.

Complete minimal working example In Listing 16 you can see a complete minimal working example. Like with the complete minimal working example above for `natbib`, the `.bib` file is bundled into this example only so that you can copy and paste it into an editor and try compiling it yourself. But this is *not* how you should write your `.tex` files.

You should instead use a single master bibliography file as discussed in §3.13.2. In this case, you would replace `\addbibresource{\jobname.bib}` with `\addbibresource{master.bib}`, if, for example, you named your `.bib` file `master.bib`. Moreover, you would also then completely delete everything from `\begin{filecontents}` to `\end{filecontents}`.

A second thing to note about the code in Listing 16—particularly in contradistinction to the example for `natbib`—is that the bibliography file is added in the preamble of your document and you *must* specify the file extension (`.bib`).

You can have multiple calls to `\addbibresource` in your preamble if you want to use multiple `.bib` files. You cannot pass the names of multiple `.bib` files to one call of `\addbibresource`; separate calls are necessary.

Finally, you print your bibliography at the end of your document by writing `\printbibliography`.

Copy and paste the example code in Listing 16 into your editor, and try compiling it using the three compilation steps discussed above.

```
\documentclass{article}

\begin{filecontents}{\jobname.bib}
@book{chomsky1995:MP,
    Address = {Cambridge, MA},
    Author = {Chomsky, Noam},
    Publisher = {The MIT Press},
    Title = {The Minimalist Program},
    Year = {1995}}
\end{filecontents}

\usepackage[
    backend=biber,% you can use bibtex instead of biber if you want
    citestyle=authoryear-comp,
    bibstyle=authoryear
]{biblatex}
% This is needed because the default authoryear style puts "In:"
% before a journal name, which is not standard in linguistics styles.
% So this code gets rid of that.
\renewbibmacro{in:}{%
  \ifentrytype{article}{}{\printtext{\bibstring{in}\intitlepunct}}}

\addbibresource{\jobname.bib}

\begin{document}

In Minimalist syntax, S-Structure has been eliminated \autocite[see][73--124]{chomsky1995:MP}.

\printbibliography

\end{document}
```

Listing 16: A complete minimal working example with `biblatex`

3.13.4 The backend processor

As mentioned above in §3.13.1, there are two backend processors that you can use for doing bibliographies with LaTeX. The purpose of the backend processor is to serve as an interface between your `.tex` document and your bibliography file (`.bib`). That is, they go through your `.bib` file and automagically pull out all of the things that you cited in your `.tex` document.

The citation package `natbib` is only compatible with the backend processor BibTeX, whereas the citation package `biblatex` is compatible with both BibTeX and Biber.

3.13.4.1 BibTeX

For a long time, the only backend processor for bibliographies with LaTeX was BibTeX. Because it was developed quite some time ago, it does not play nicely with accented Latin characters and non-Latin alphabets. For this reason, I would encourage you to use Biber with `biblatex` instead.

3.13.4.2 Biber

Biber is a modern alternative to BibTeX. In addition to playing nicely with accented Latin characters and non-Latin alphabets, another advantage of Biber is that it is capable of reading and processing many more entry types and data fields than BibTeX is (cf. §3.13.1.1 and fn. 31). This is the basis for many of the advanced and versatile features of the `biblatex` package.

3.14 Paragraphs

Before moving on to stuff that is specifically useful for linguistics, let's end this general LaTeX 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 start a new paragraph. Perhaps unsurprisingly then, many new LaTeX users will often do stuff like what is shown in Listing 17.

```
\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 17: 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 18 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 18, I've given semantics that are similar to those 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 18 yourself and see what happens.[41]

```
\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 18: Good practice for typesetting paragraphs

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.[42]

4 Useful stuff for linguists

4.1 `fontspec` and Unicode

In fn. 21, I mentioned a better way of typesetting diacritics in LaTeX. This is where the `fontspec` package comes in. Historically, using LaTeX with fonts hasn't really been a thing. This has changed recently with the advent of two other engines—XeLaTeX and LuaLaTeX—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 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 19.[43]

[41] Instead of using the two `\setlength` commands in Listing 18, I would actually just recommend loading the `parskip` package. It gives you handout-y semantics for paragraphs in a much better and more robust way. Listing 18 is only given for pedagogical purposes—namely, to illustrate the point of semantic markup for paragraphs, which isn't necessarily clearly illustrated by just loading a package. Nonetheless, just loading the `parskip` package instead is indeed the better way to go.

[42] Explaining what exactly a version control system (VCS) is, is quite beyond the scope of this document. 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.

[43] See also this question and its answers on TeX.SX. Note that in order for the example in Listing 19 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

```
\documentclass{article}
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{Times New Roman}
\newfontfamily\phonetic[]{Doulos SIL}
\usepackage{textglos} % good semantic markup for inline examples, like \xv{...}, \xm{...}, etc.
\begin{document}

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

\end{document}
```

Listing 19: Example of using a distinct font for phonetics

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 TEX. 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 20, rather than having to use a separate font for special (phonetic) characters.

```
\documentclass{article}
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{CMU Serif Roman}
\usepackage{textglos} % good semantic markup for inline examples, like \xv{...}, \xm{...}, etc.
\begin{document}

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

\end{document}
```

Listing 20: Using one font that has a lot of Unicode glyphs

Processing either Listing 19 or 20 with either LuaLATEX or XƎLATEX 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 *very 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 this document 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`.[44] 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`.

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.

[44] 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 LATEX (on the Mac)*, its markup is not all that semantic.

4.2.1 Basic linguistic example

Listing 21 gives an example of how to typeset some basic examples.[45] 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.

```
\documentclass{article}

\usepackage{textglos} % good semantic markup for inline examples, like \xv{...}, \xm{...}, etc.

%\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 21: Typesetting basic linguistics examples

4.2.2 Glossing examples

With `gb4e`, you can also gloss examples. An example is shown in Listing 22. The package `leipzig` provides several feature glossing macros, following the glossing standards for the field of linguistics as set forth by the Leipzig Glossing Rules. You can find a list of all the macros that `leipzig` provides in the package documentation. Most of them are pretty straightforward to guess, like `\Acc{}` for accusative and `\Prog{}` for progressive; however, some of them you might need to look up in the package documentation.

While the Leipzig Glossing Rules are relatively comprehensive, you might find yourself in a situation where you need to use a gloss that is not standardized in the Leipzig Glossing Rules. If this need arises, `leipzig` provides a way to define your own glossing macros. One further feature of the `leipzig` package that is really nice is that it provides

[45] 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.

```
\documentclass{article}
\usepackage{gb4e}
\noautomath
\usepackage{leipzig}
\begin{document}
\begin{exe}
    \ex[]{\gll Der Apfel würde gegessen.\\
    The.\M{}.\Sg{}.\Nom{} apple was eaten\\
    \trans `The apple was eaten'
    }
\end{exe}
\end{document}
```

Listing 22: A glossed example with `gb4e`

a way, via the `glossaries` package, to automagically typeset a list of all glosses used in a paper along with their definitions. This is something that is very commonly printed in a footnote in a paper the first time a gloss is used in that paper. Listing 23 gives an example of both of these features of the package.

In order to correctly compile Listing 23, there are some special compilation steps that you will need to go through, just like with bibliographies. This is because the `glossaries` package needs to make use of some information written to the `.aux` file after you run `latex` for the first time. The compilation steps that you will need to go through in order to get this to work are the following three steps:

(i) `latex myfile.tex`

(ii) `makeglossaries myfile`

(iii) `latex myfile.tex`

The second command runs a Perl script that should have been installed when you installed your TeX distribution. Notably, you should just call the script on the base name of your file; do not specify a file extension when you run the command `makeglossaries`! You should also ensure that your file name does not include spaces, which might give the `makeglossaries` script a hard time. Just like with bibliographies, it is possible to run these compilation steps from the command line, but it's also possible to configure your editor to run the `makeglossaries` command. And again, `latex` should be treated as a placeholder for whatever engine you are using. For example, if you are using XeLaTeX, then you should replace `latex` with `xelatex` in two of these three steps.

```
\documentclass{article}
\usepackage{gb4e}
\noautomath
\usepackage{leipzig}
\makeglossaries
\newleipzig{pol}{pol}{polite}
\begin{document}
As can be seen in (\ref{ex:causative}), \ldots
\footnote{The following abbreviations are used in this paper: \printglosses}

\begin{exe}
    \ex[]{\gll Hasan geu-peu-reubah aneuk miet nyan\\
    Hasan \Third{}\Pol{}-\Caus{}-fall child small \Dem{}\\
    \trans `Hasan caused the child to fall'
    }
    \label{ex:causative}
\end{exe}
\end{document}
```

Listing 23: An example with `leipzig` and `glossaries`

There are a few further things worth noting about the use of `leipzig` package. In order for you to be able to automagically print out a list of all glosses used in the paper and their definitions, you must execute `\makeglossaries` in the preamble, after loading the `leipzig` package. This then allows you to use the command `\printglosses` to print out all glosses used in the document. Moreover, if you need to define some glosses that are not part of the Leipzig Glossing Rules, such as a polite feature gloss as shown in the example, you can do so with `\newleipzig`. New feature glossing macros should be declared after `\makeglossaries`.

`\newleipzig` takes three arguments. The first argument is a label, which is what will be used to define the macro, except with the first letter being capitalized. As can be seen in Listing 23, the first argument to `\newleipzig` is `pol`, which means that the command that will be created for you to use inside the document itself will be `\Pol{}`. The second argument is the short form of the abbreviation. This is what will show up in small caps in the gloss itself. In Listing 23, the second argument to `\newleipzig` is `pol`, which means that POL is what will show up in the gloss. Note that this second argument must be all lowercase. Finally, the third argument is the definition of the gloss that will show up when you print out the abbreviations and their definitions with `\printglosses`. In Listing 23, the third argument to `\newleipzig` is `polite`.

One last thing that is worth noting about the `leipzig` package is that it should be loaded after you load the `hyperref` package, if you are also using that package. Moreover, `hyperref` should generally be the last package that you load in your preamble, so this means that the last two packages you will probably want to load are `hyperref` and `leipzig`, in that order.

If you haven't already, try typesetting the example in Listing 23 yourself, to make sure that you've got the hang of doing the three different compilation steps. Note that if your document has a bibliography, you will still need to do the compilation steps for the bibliography as well, so your compilation process might ultimately involve five steps:

(i) `latex myfile`

(ii) `biber myfile` OR `bibtex myfile`

(iii) `makeglossaries myfile`

(iv) `latex myfile`

(v) `latex myfile`

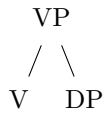
To make sure you've further got the hang of the `leipzig` package, try defining a new gloss that isn't part of the standardized set from the Leipzig Glossing Rules.

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`.[46] The syntax for both of these is almost the same, except that `forest` is a bit more forgiving in its syntax for labeling the leafs in a tree.

Let's start with a very basic example, given in Listing 24, which will produce what you see in (1).

Try typesetting this yourself.

(1)


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 in turn makes it easier to debug if you are getting errors when trying to compile your file.

[46] For further discussion, see this question and its answers on TeX.SX.

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

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

\end{document}
```

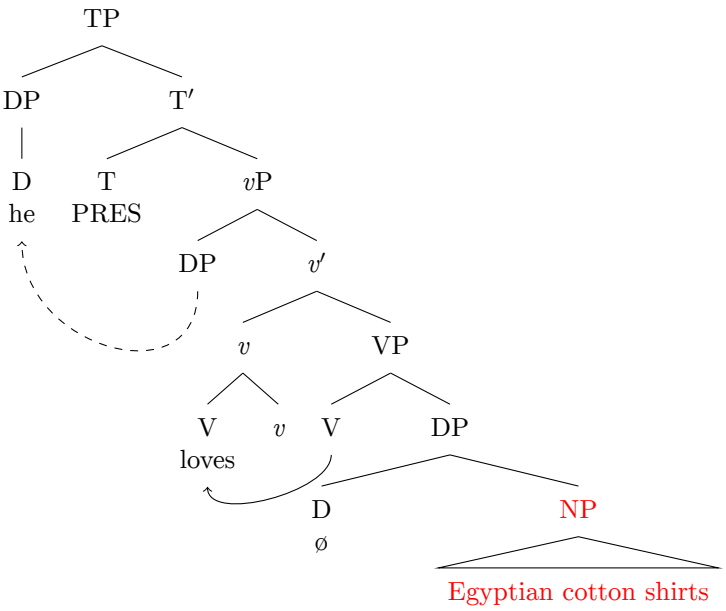
Listing 24: A very basic example with `forest`

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. 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 25, which will address some of these issues. The code in Listing 25 will produce what you see in (2).

(2)



A few comments about what you see in Listing 25 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 the 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 25.

```
\documentclass{article}
\usepackage{forest}
\useforestlibrary*{linguistics}
\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, for c-commanded={red}]
                                [NP
                                    [{Egyptian cotton shirts}, roof]
                                ]
                            ]
                        ]
                    ]
                ]
            ]
        ]
        \draw[->] (V1) [in=-90, out=-90, looseness=0.75] to (V2);
        \draw[->,dashed] (DP1) [in=-90, out=-90, looseness=1.5] to (DP2);
        \end{forest}
    }
\end{exe}
\end{document}
```

Listing 25: An example of typesetting a syntax tree using `forest`

Finally, you will most likely want all of your trees to be typeset using the default styles of the `linguistics` library for `forest`. The `linguistics` library was introduced in version 2.0 of the `forest` package. If you have an outdated version of `forest`, I would highly recommend upgrading to the latest version.

By using the `linguistics` library for `forest`, three default styles are applied to all trees: (i) the branches (lines) that connect leafs in a tree are set to meet at the bottom of their parent leaf; (ii) the baseline of the resulting tree is set to the top of the tree, which causes the top of the tree to align with the example number when the tree occurs inside of a numbered example (rather than aligning with the bottom of the tree); and (iii) leafs that span multiple lines are set to have centered horizontal alignment.

If you'd rather that these three styles were not applied to every tree by default, you can load the `linguistics` library with the unstarred variant of the `\useforestlibrary` command (*i.e.*, `\useforestlibrary{linguistics}`). Then, to apply the styles to any given tree, you would need to add the three styles to each tree before writing the bracketed structure for the tree, like the following:
`\begin{forest} sn edges, baseline, for tree={align=center} [VP [V] [DP]]\end{forest}`.

The package `forest` also defines several other handy options, the first two of which are exhibited in Listing 25:

- `roof`, for drawing triangles
- `c-commanded`, for visiting all leafs c-commanded by the current leaf
- `c-commanders`, for visiting all leafs that c-command the current leaf

These options are intended to be used at a particular leaf and do not apply to the tree as a whole.

There are also two other styles that `forest` defines. Styles are applied to a tree as a whole and are written before the bracketed structure of the tree. These two styles are `nice empty nodes` and `GP1`. The style `nice empty nodes` can be used when you have leafs without labels and you want the branches of child leafs to connect to the branch of the parent leaf. The style `GP1` can be used to typeset Government Phonology representations. If you'd like to see examples of both styles, you can take a look at the `forest` documentation.

4.4 Typesetting OT tableaux

There are two options that I would recommend for typesetting OT tableaux. One package is called `OTtablx`, 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 TEX distribution can see (*i.e.*, `texmf/tex/latex`; cf. §2.3).[47]

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 TEX distribution, such as TeX Live or MiKTEX). An example of how to typeset a tableau is given in Listing 26. This code will produce the result in (3).[48]

(3)

/bad/	*Voiced-Coda	Ident-IO(voice)	*[+voi,–son]
a. bad	*!		**
☞ b. bat		*	*
c. pat		**!	
d. pad	*!	*	*

5 Things to learn on your own

This handout is already getting much too long. Moreover, one of the best ways to get comfortable with LATEX (or anything for that matter) is to try learning some stuff on your own.

Here are some packages or tools (and some brief descriptions thereof) that I would highly recommend exploring on your own at some point.

[47] Note that if you do decide to try `OTtablx` 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 XELATEX, things should work fine, but it will not work with pdfLATEX. See this question and its answers on TeX.SX for discussion.

[48] Disclaimer: I'm not a phonologist.

```
\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 26: An example of an OT tableau using `ot-tableau`

`latexmk` This is a tool that tries to automagically guess what compilation steps you need to do in order to successfully compile your document. It will then try to do all of these compilation steps for you. If you get tired of going through all of the compilation steps necessary for doing a bibliography in LaTeX you might try compiling your document with `latexmk`.

`arara` An alternative to `latexmk` that I would highly recommend is `arara`. Rather than trying to automagically guess the compilation steps necessary, `arara` allows you to specify the compilation steps that are necessary at the top of your document using a special syntax. Of course, this requires knowing which compilation steps are necessary, so it's not quite as straightforward as `latexmk`. However, it allows for much more fine-grained control of the compilation process that might be preferable when you start using more advanced tools.

`datatool` The `datatool` package allows you to loop over information stored in a `.csv` file and incorporate it into your `.tex` document.

Different document classes Try exploring some different document classes. In this handout, we've only used the basic `article` document class. If you want to use LaTeX for presentations, take a look at the `beamer` document class. Another document class that is useful for typesetting longer documents, such as a monograph, is the `memoir` document class.

`knitr` If you use R for statistics and processing data, I would *highly* recommend looking at `knitr`. This is a package that will allow you to write R code and LaTeX markup in the same source document. If you do this, then you can automagically generate things like p-values for inclusion in your document, for example.

6 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 to 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.

If you're struggling with a particular package, try reading the documentation. Unlike most open-source software projects, LaTeX 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 both TeX Live and MiKTeX. You can just open a terminal and type `texdoc <package name>`.

And, of course, there are also some good books for learning LaTeX. There's a free one called the *The Not So Short Introduction to LaTeX 2ε*.[49] There's also *The LaTeX Companion*.

7 Acknowledgments

Many thanks to both Alan Munn and Kenneth Hanson for reading over early drafts of this handout and providing useful feedback and suggestions. Thanks also to Joseph Wright for help with sorting out the problems with the previously untested and broken commands in Listings 2, 4, and 5. Moreover, I also owe many thanks to TeX.SX for being a great resource, for helping me to learn a lot about LaTeX, and, most importantly, for being a great community consisting of a bunch of friendly folks.

8 Feedback

If you have comments or suggestions, please feel free to either email me, open a GitHub issue, or submit a pull request with a suggested change to this document.

[49] If you have a TeX distribution installed, you can find this book on your system by typing `texdoc lshort` at a terminal.