LUCL LaTeX Workshop—Share`LaTeX`

Martin Kroon
`m.s.kroon@hum.leidenuniv.nl`

9th May 2018

Today we will make a document in LaTeX. We will use the website Overleaf for this, which is kind of like Google docs for LaTeX.

Why do we want to use LaTeX? First of all, this here document was made using LaTeX. Looks pretty, right? Also:

- anything is possible in LaTeX;
- you can define your own commands and macros;
- making a bibliography is very easy;
- good support for other scripts, graphs, tables, equations, trees, etc.;
- it allows for modular documents;
- and you can make nice lists like these, among other things.

I will admit, LaTeX's learning curve can be quite steep; that's why today, I will walk you through all the basics.

1 Making a document

In your Overleaf "Projects" view, click **New Project** to start a new project. Click "Blank Project" for today—in a later stadium you can also use a project template, upload a project from your computer or import a project from GitHub (for those who are familiar with it). You are then prompted to give your project a name—think of something, but you can of course always change it later on.

1. Make a project and call it `LaTeX workshop`.

Once you've created a project, you will end up in the editor. The editor consists of three parts. **Left** is the project overview: here we see all files in your project. As noted above you can have modular projects, where you import files into other files. For now, we have only one file: `main.tex`. **In the middle** we have the text editor, where we write our article, book or even letter or exam—this is the **source code**. On the **right** we have a PDF view of your current

project, so that you can check if everything is going according to plan. You can click **Recompile** to make Overleaf update to PDF into its latest version (or press Ctrl+Enter or even Ctrl+s).

Compiling is done by a backend compiler. By default Overleaf uses pdfLaTeX to interpret your LaTeX code and turn it into a PDF. You usually don't have to worry about this, but linguist have the nasty habit of using weird characters every now and then, which upsets pdfLaTeX. For this reason we advise you to use XeLaTeX instead. You can switch compilers by clicking the Menu button in the top left corner.[1]

2 Document classes and packages

Our `main.tex` looks like this:

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

\title{LaTeX workshop}
\author{<your name>}
\date{May 2018}

\begin{document}

\maketitle

\section{Introduction}

\end{document}
```

All lines with text start with a `\`. In LaTeX, functions (or **commands**) start with a `\`. Required arguments are passed to commands between curly brackets (`{...}`), optional arguments are passed to commands between square brackets (`[...]`) and come between the command and its required arguments.

In more detail the first line defines the document's class. By default this is `article`, but we can also make a `book`, `letter` or even a presentation. This has some influence on the layout of the document.[2] The second line imports a **package** (with the optional argument `utf8`). Other packages can for example allow you to import images, to use IPA or to customize headers in detail. We'll come back to some packages later on.

An important remark about the document structure: a LaTeX document has a head (called the **preamble**) and a body. The body is everything between

[1] If you choose to do so, it is good practice to remove the line with `\usepackage[utf8]{inputenc}`, which is rendered unnecessary by XeLaTeX.

[2] For paper submissions, journals and conferences often distribute a style file for you to use when writing your paper—strictly speaking they define their own document class in which they define a whole bunch of stuff, such as page layout and bibliography style.

`\begin{document}` and `\end{document}`. Everything before that is the preamble. In the preamble we define everything needed to make our document: we load necessary packages here, define metadata and define the document's class.

The command `\maketitle` prints a title using the metadata defined in the preamble. The style of the title differs per document class—e.g. an article prints it on the first page, a book makes a dedicated title page. You can also customize the style of your title later on using packages such as `titling` or `titlepages`.

The bare necessities for a LaTeX file are a `\documentclass{...}` line in the preamble and a body defined with `\begin{document}` and `\end{document}`. Everything else is in theory optional.

2. Import the package `graphicx`, for we will need it later on.

3 Basic text editing

Another important remark is that LaTeX does not interpret a newline as a new line, whereas Word does. This means that if I type

```
John has a cat.
The cat is also called John.
```

LaTeX will print these two sentences on the same line. In order to force a newline, type \\. So

```
John has a cat.\\
The cat is also called John.
```

will yield two different lines. However, they are still the same paragraph. You can start a new paragraph by leaving a empty line between two bits of text. E.g.

```
John has a cat.

The cat is also called John.
```

will yield two paragraphs with indentation. Try it out to see the difference.

Yet another important remark is LaTeX's behaviour concerning quotation marks. Whereas Word automatically changes quotation marks to their right form (i.e. facing left or facing right), LaTeX does not, unfortunately. In LaTeX, single left quotation marks are ` (grave accent), single right quotation marks are ' (acute accent). Double quotation marks are printed by simply doubling quotation marks. E.g.

```
``John'' or ''John''
```

which yields “John” or ”John”. the first one is correct.

Italics and **boldface** are instantiated by `\textit{...}`, and `\textbf{...}`, respectively. Instead of typing the commands every time, Overleaf supports the

shortcuts Ctrl+i and Ctrl+b. For SMALL CAPS, `typewriter` and sans serif styles, use `\textsc{...}`, `\texttt{...}` and `\textsf{...}`, respectively.

Font size is also controlled with macros. This means they don't take arguments. By calling any of the commands in Table 1, all text coming after it will be printed in that size.[3] In order to change back to how it was, just call `\normalsize` again. Alternatively, you can make it so that the font-size change is limited to a certain portion of text, say one word. You can do this by encapsuling the font-size command and the word between curly brackets, like so:

```
{\large John} has a cat.
```

`\tiny`	sample text
`\scriptsize`	sample text
`\footnotesize`	sample text
`\small`	sample text
`\normalsize`	sample text
`\large`	sample text
`\Large`	sample text
`\LARGE`	sample text
`\huge`	sample text
`\Huge`	sample text

Table 1: Font sizes with a sample text.

3. Play around with newlines, italics, boldface and font sizes a bit to acquaint yourself with it. Also try out the quotation marks.

4 Commenting out

If there is a line that you don't want in your final PDF but you don't want to delete it, you can tell LaTeX to ignore it when it's compiling—this is called **commenting out**. In LaTeX this is done with a `%`: putting one in front of a line will have the line be ignored. You can also press Ctrl+?.

If you want, though, a simple percentage sign in your text, you need to type `\%`, otherwise LaTeX will interpret it as a comment.

[3] To change the font size of your entire document, you can do so by adding `12pt`, `14pt` etc. as an optional argument to `\documentclass`. LaTeX will also change the page margins accordingly for you. For instance, `\documentclass[14pt]{article}` will create an `article`-type document with a font size of 14pt. The LaTeX default is 11pt.

5 Diacritics

Relevant for us linguists, using diacritics in LaTeX is substantially easier than in Word, because it is also done using commands.

Do you want a circumflex on a c-cedilla, or a long Hungarian umlaut on top of a q? No problem: ç̂ q̋ (`\^{\c{c}} \H{q}`). A list of shortcut commands for diacritics can be found in the table below. If you are using XeLaTeX (see Section 2), you can also enter signs + diacritics (ó, ą, š etc.) directly using your keyboard.

Description	**LaTeX command**	**Output**
Grave accent	`\`{o}`	ò
Acute accent	`\'{o}`	ó
Circumflex	`\^{o}`	ô
Umlaut/trema/dieresis	`\"{o}`	ö
Double acute	`\H{o}`	ő
Tilde	`\~{o}`	õ
Cedilla	`\c{o}`	o̧
Ogonek	`\k{o}`	ǫ
Barred L	`\l{}`	ł
Macron	`\={o}`	ō
Bar under the letter	`\b{o}`	o̲
Dot over the letter	`\.{o}`	ȯ
Dot under the letter	`\d{o}`	ọ
Ring over letter	`\r{a}`	å
Breve	`\u{o}`	ŏ
Caron/háček	`\v{o}`	ǒ
Tie over two letters	`\t{oo}`	o͡o
Slashed o	`\o`	ø
Dotless i or j	`\i{}` or `\j{}`	ɪ or ȷ

Table 2: Diacritics in LaTeX

Putting diacritics on capital letters works in the same way, and you can even combine multiple diacritics if your font allows this: n̰̄ (`\={\d{n}}`).[4]

4. Try to reproduce the following (non-sensical) combinations: p̋, ḡ, i̯, *̇.

6 Document structure and table of contents

Making a table of contents in LaTeX is substantially easier than in Word as well. For this we can use LaTeX's built-in sectioning commands.

[4]The standard font provided by LaTeX is not well-suited for this. As detailed here https://tex.stackexchange.com/questions/159291/multiple-diacritics-on-one-character, freely downloadable fonts such as *Brill* or *SIL Charis* work perfectly.

In our `main.tex` file, we started of with a line saying `\section{Introduction}`. This command prints a section header with the title *Introduction*. But apart from that, LaTeX keeps track of every section title and on which page it starts for later to use in the table of contents. LaTeX also automatically numbers sections (which you can later automatically reference: see Section 10 for referencing things).

There also exist commands for subsections and sub-subsections, chapters and parts. They all have an internal level: sectioning commands with higher levels are embedded in sectioning commands with lower levels (so a chapter, with level 0, can contain sections, with level 1; sections cannot contain chapters, etc.). If necessary you can also define your own sectioning command on a new level (for instance if you for some reason need a sectioning command that can contain parts). In case you do not like the style of a sectioning command, you can customize them using the package `titlesec`.

Command	**Level**
`\part`	-1
`\chapter`	0
`\section`	1
`\subsection`	2
`\subsubsection`	3
`\paragraph`	4
`\subparagraph`	5

Table 3: Sectioning commands.

Then to make a table of contents, just type `\tableofcontents` wherever you want it printed (for instance directly after your `\maketitle`. A dedicated table of contents for figures and tables can also be instantiated using the command `\listoffigures`: more on figures and tables in Sections 8 and 9.

Two more things. Sectioning commands can take an optional argument (so between the command and the curly brackets) which is the title of the section (or chapter etc.) as it should be printed in the table of contents. Then, if you don't want a section (or chapter etc.) to receive a number, you can put a `*` between the command and the curly brackets. However, this will also make that it is not in the table of contents![5] So:

```
\section{A}
This is section 1, called A.

\section[B]{C}
This is section 2, called C in the header, but B in the TOC.
```

[5]It being LaTeX and anything being possible, if you don't want a section a number but you do want to have it appear in the table of contents, it is possible using the command `\addtocontents{toc}{...}`

```
\section*{D}
This is an unnumbered section, called D.
It won't show up in the TOC.
```

5. Play around with sectioning commands and the table of contents a bit to acquaint yourself with it.

7 List environments

Making lists with bullet points or enumerated lists can be done by initiating a list environment.

Environments in LaTeX are delimited blocks, within which everything will be formatted in a special manner depending on the environment. Environments are characterized by their `\begin{...} ... \end{...}` syntax. We have actually already seen an environment, the `document` environment.[6]

Enumerated lists (or ordered lists) are introduced by the `enumerate` environment, bullet point lists (or unordered lists) by the `itemize` environment. Items inside list environments are then introduced with `\item`. Like this:

```
\begin{enumerate}
    \item The first item.
    \item The second item.
\end{enumerate}

\begin{itemize}
    \item Another item.
    \item Yet another one.
\end{itemize}
```

Lists can be embedded inside other lists.

For ordered list, the first level is by default enumerated using Arabic numbers, the second level with lowercase letter, the third level with roman numerals, and the fourth with uppercase letters. This can be changed.

Unordered lists are itemized by bullets, dashes, asterisks, and dots in the first, second, third and fourth level, respectively. This can also be changed. You can also have one item in the list have a different icon by giving the `\item` command an optional argument, e.g. `\item[$\square$]`.

6. Make a list and add a few embedded lists in it. Mix ordered and unordered lists and give one or two items another icon or label.

[6] Strictly speaking, the trick we used to only apply the large font size to *John* in Section 3 was also an example of an environment: the `\large` command was only applied within the delimited area.

8 Importing an image and figures

You can import images using the `graphicx` package we already loaded.

7. Upload an image file to your project with the "Upload" button in the top left of your Overleaf screen. Supported image files are PDF, JPEG and PNG (with `graphicx` at least).

Once you uploaded your file, you can insert it with the command `\includegraphics{...}`. If the image is a bit too big or too small, you can adjust its scale, width and height in a keyword argument. Personally I often adjust the size of images to (a fraction of) the width of the text. Invoking the following command prints the university's logo to half the width of the text:

```
\includegraphics[width=.5\textwidth]{LU.png}
```

Universiteit Leiden

However, as you can see, this image is not aligned to the middle of the page, nor does it have a caption or can we reference it. In order for all this to be possible, we can use the `figure` environment. When using the `figure` environment LaTeX also figures out itself where to place the image so that it fits nicely on the page. For example:

```
\begin{figure}[ht]
    \centering
    \includegraphics[width=.5\textwidth]{LU.png}
    \caption{This is the university's logo.}
\end{figure}
```

The `[ht]` will have LaTeX try to print the figure 'here' (`h`), and if that is not possible to print it at the top of the page (`t`). Other possible positioning options are `b` (bottom of the page) and `p` (which will print the image on a dedicated page for figures).

8. Try inserting an image using the `figure` environment. Also try a few positioning options and try adjusting the size of the image.

9 Making a table

Tables in LaTeX take some getting used to and may not always be very clear, but they are not very difficult. Let's have a look.

Tables work in a very similar fashion as figures: they also use an environment in which the actual table is formatted. For example:

```
\begin{table}[ht]
    \centering
    \begin{tabular}{|r|cl|}
        \hline
        Country & Capital & Inhabitants \\\hline
        Netherlands & Amsterdam & 17.02 M\\
        Germany & Berlin & 82.67 M\\\hline
    \end{tabular}
    \caption{A table about the Netherlands and Germany.}
\end{table}
```

In the above example, we see that the `table` environment is invoked, in which the `tabular` environment is invoked. The `tabular` environment is the actual table, the `table` environment is where the table's position and caption are controlled. You can also make a `tabular` outside a `table`, just like you can insert an image outside a `figure`.

In the `tabular` environment, `|r|cl|` means that there are three columns, first of which is right-aligned (`r`), second of which is centred (`c`) and third of which is left-aligned (`l`). The pipes (`|`) tell LaTeX where to put vertical lines: so, there are vertical lines in the table on the left, between the first and second column and on the right. Horizontal lines are printed with `\hline`. Then columns are separated by `&`, rows by `\\`. So, the above example gives us:

Country	Capital	Inhabitants
Netherlands	Amsterdam	17.02 M
Germany	Berlin	82.67 M

Table 4: A table about the Netherlands and Germany.

You can merge cells in a table with the command `\multicolumn{...}{...}{...}` or using the package `multirow`. See https://en.wikibooks.org/wiki/LaTeX/Tables#Spanning.

9. Try making a table.

You can also use the application on the following website to automatically generating a LaTeX table with a clear user interface: https://www.tablesgenerator.com/.

9.1 Advanced table stuff

Here are a few packages which make your tables look even fancier.

9.1.1 `booktabs`

As an alternative to the command `\hline`, this package allows you to create horizontal lines in your table using the commands `\toprule` (for the top line of your table) `\midrule` (after the header) and `\bottomrule` at the bottom of your table). The results can be seen in Table 2 (above). Vertical lines do not go well in combination with the horizontal lines drawn by `booktabs` (try it out and see why!), but the use of vertical lines in tables is deprecated in typographic style guides anyway.

9.1.2 `ltablex`

LaTeX encounters some difficulties when cell contents or tables themselves are bigger than the page they are supposed to fit on, but these are all easily to solve! For tables bigger than one page, the package `longtable` was invented; for tables containing cells with a lot of text, there is `tabularx`. These two packages are conveniently combined into one neat package, `ltablex`. Check out the following example.

The command `\endhead` means: all the lines above this command need to be repeated on every new page. In addition, everything above the command `\endfoot` is repeated at the bottom of the page whenever LaTeX has to put a table on multiple pages. Lastly, everything above `\endlastfoot` is only printed at the very end, which is why we put our `\caption{...}` there.

Cell one	**Cell two**
Very long text to be put in the first cell	Even longer text that should be put in the second cell and which would definitely go over the edge if it wasn't for `tabularx`
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line

Continued on next page

Cell one	**Cell two**
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line
Another line	Yet another line

Table 5: Testtable

```
    \begin{tabularx}{\linewidth}{XX}
    \toprule
    \textbf{Cell one} & \textbf{Cell two} \\
    \midrule
    \endhead

    \bottomrule
    & \hfill \textit{Continued on next page} \\
    \endfoot

    \bottomrule
    \caption{Testtable}\label{tbltest}
    \endlastfoot

    Very long text to be put in the first cell & Even longer text
    that should be put in the second cell and which would definitely
    go over the edge if it wasn't for \texttt{tabularx} \\

    Another line & Yet another line \\
   ...
    Another line & Yet another line \\
\end{tabularx}
```

Note that this table does not need a `\begin{table} ... \end{table}` environment or the command `\centering` to keep the table on the center of the page. It does all of that on its own!

Basically, a `tabularx` says: I want to make a table which is as wide as the line itself (`\linewidth`), and I have two columns for this (the two X'es in the second obligatory argument). The use of X as a column type here means: "You try to find the best size for this cell, as long as the resulting table stays within the size of `\linewidth`." Generally, LaTeX will try many different cell sizes until it finds a good one which fits the contents. This makes sure that the table will never go wider than the width you set for it, but it takes a longer time to compile.

Note that you can change `\linewidth` in the first obligatory argument if you want a different size limit (for instance, to '50mm'). You can also combine X columns (with no fixed size) with the regular `l`, `c` and `r` columns mentioned before.

9.1.3 `threeparttable`

This package allows you to add table notes at the bottom of your table. An example together with its code are given here.

Spring	**Summer**	**Fall**	**Winter**
Hazel trees	Plum trees[1]	Apple trees	Money trees[2]

[1] Chestnuts are great.
[2] Note sure if these are real though.

Table 6: Trees I like

```
\begin{table}[ht]
   \centering
  \begin{threeparttable}[h!]
   \begin{tabular}{llll}
          \toprule
          \textbf{Spring} & \textbf{Summer} & \textbf{Fall} & \textbf{Winter} \\
          \midrule

          Hazel trees & Plum trees\tnote{1} & Apple trees & Money trees\tnote{2} \\

          \bottomrule
    \end{tabular}
    \begin{tablenotes}
        \item [1] Chestnuts are great.
        \item [2] Note sure if these are real though.
    \end{tablenotes}
    \caption{Trees I like}\label{tbltrees}
  \end{threeparttable}
\end{table}
```

Note that this table consists of three parts (hence the name): there is a `table` environment which contains 1. a `threeparttable` environment, which contains 2. a regular `tabular` and immediately after that 3. a `tablenote` environment. You can put notes in your table where ever you want with `\tnote{#}`, and refer to these notes with `\item [#]` in the `tablenote` environment. Just add `\usepackage{threeparttable}` to your preamble and you're good to go![7]

[7] To change the size of the table notes, just copy and paste the following three lines into

10. Replicate the following table, using `booktabs` and `threeparttable`.

Berlin	**London**	**Amsterdam**
Anna[1]	Colin	Laura
Helmut	Sarah	Rogier[2]

[1] Smart girl; knows LaTeX.
[2] Still owes me money!

Table 7: My friends

10 Referencing sections and figures

You can automatically reference sections, chapters, tables, figures, equations, etc. by giving them a label. For example, this section was referenced earlier on, and in order to do that we gave this section a label:

```
\section{Referencing sections and figures}\label{refs}
```

We referenced its number with `\ref{refs}`.

For sectioning commands and items in a list, just put the label anywhere inside the section or item. Tables, figures and equations you can give a label inside the environment.

You can also reference the page on which something is or starts using `\pageref{...}`.

```
This section starts on page \pageref{refs}.
```

yields

This section starts on page 13.

11. Try referencing the table you just made.

11 Bibliography management in LaTeX

Bibliography management and citing references in LaTeX is very simple. The gist of it is that you have one file containing all your references and then in your main text you cite them with specific commands. You can also very easily change the citation style or correct an error in a reference. There are three main

your preamble:
```
\makeatletter
\g@addto@macro\TPT@defaults{\footnotesize}
\makeatother
```

options in LaTeX to manage your bibliography. Today we will use `biblatex`, but do also check out the other possibilities, `bibtex` and `natbib`.

The file containing all your references is a `.bib` file. For every reference in there, you define its type and give all metadata. For example:

```
@article{lander1966counterexample,
    title={Counterexample to {E}uler's conjecture on sums of like powers},
    author={Lander, L. J. and Parkin, T. R. and others},
    journal={Bull. Amer. Math. Soc},
    volume={72},
    number={6},
    pages={1079},
    year={1966}
}
```

Here we define with `@article` that the reference is an article. What follows, `lander1966counterexample`, is the reference key. With this key you cite this reference in your text. Make sure your `.bib` file does not contain any duplicate keys. Then we define the metadata, such as title, author and the name of the journal the article was published in. Different reference types can have different metadata: check https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management#biblatex. It is okay not to include all metadata: if you don't know the volume of the journal it was published in, don't put it in the `.bib` file, `biblatex` takes care of it when citing it.

As some of you may know, you can use Google Scholar to cite something. What you may not know is that Google Scholar also gives you the option to cite something as a `biblatex` reference!

One note on capital letters: `biblatex` changes everything to lowercase, except for the first letter. So, if you want to have something with a capital that isn't the first letter, put curly brackets around it. We also did it in the example above for *Euler*.

12. Make a `.bib` file in your Overleaf project, go to Google Scholar, and get a citation in `biblatex` style. Put it in your `.bib` file.

In order to cite your reference in your text, you need to import a few packages and define a few things. First we need to load `biblatex`-today we shall want it to cite in APA and use a specific back-end (you can ignore this for today). Second we need to define the language mapping for `biblatex` (because it also supports citing in other languages!), but for that we first need to load a language package: in this case `babel`. Lastly, we need to tell `biblatex` where to find the references: in your `.bib` file. This is all done in your preamble. Like this:

```
\usepackage[british]{babel}
\usepackage[style=apa,backend=biber]{biblatex}
\DeclareLanguageMapping{british}{british-apa}
\addbibresource{<your .bib file>}
```

Now we can cite references. For in-text citations, use `\textcite{...}`. For parenthetical citations, use `\parencite{...}`. When you're citing multiple references at once, just pass multiple reference keys as arguments at the same time, separated by a comma (no space!). If you want to add things like "see" or "pp. 48-49", you can do that as optional arguments:

```
\parencite[see][p. 48]{lander1966counterexample}
```

yields

(see Lander, Parkin et al., 1966, p. 48)

Then in order to print your bibliography, just type `\printbibliography` wherever you want the bibliography.

13. Add the lines above to your preamble, and try citing your reference in your text. Also try to change the citation style to MLA to see what happens.

12 Linguistic packages

Let's also have a quick look at how to use IPA and how to gloss in LaTeX.

IPA is taken care of by the package `tipa`. It defines several character mappings. Check out https://www.tug.org/tugboat/tb17-2/tb51rei.pdf for a full overview of all character mappings. For example, `\textipa{f@"nEtIks}` yields fəˈnɛtɪks.

The package `gb4e` takes care of linguistic examples and glossing. Examples are formatted in a similar environment to lists. For example, where the second example is how you format ungrammatical sentences:

```
\begin{exe}
    \ex This sentence is grammatical English.
    \ex[*] {This sentence English in ungrammatical is.}
\end{exe}
```

This yields the following. Note that, just like lists, examples keep track of the counter themselves:

(1) This sentence is grammatical English.

(2) * This sentence English in ungrammatical is.

In order to gloss an example, you need two more commands inside the `exe` environment: `\gll` (glossing) and `\trans` (translation). For example the Finnish sentence:

```
\begin{exe}
    \ex \label{finnish}
    \gll Pekka pel\"{a}sty-i karhu-sta.\\
         Pekka {get scared}-PST bear-ELA\\
    \trans `Pekka got scared because of the/a bear.'
\end{exe}
```

yields

(3) Pekka pelästy-i karhu-sta.
Pekka get scared-PST bear-ELA
'Pekka got scared because of the/a bear.'

Note that `get scared` is between curly brackets: this is to treat is as one unit. The spaces are normally parsed as separators.

You can also reference examples using the `label-ref` system, just like items in a list.

If you want more information on linguistics and LaTeX, do check out https://en.wikibooks.org/wiki/LaTeX/Linguistics.

13 Defining macros and commands

Since LaTeX is a programming language, you can define your own macros and commands. In programming, it is generally true that if you have to do something twice, you are doing it wrong. Macros are the same as commands, but don't take any arguments.

Say we want to make a macro for the word *supercalifragilisticexpialidocious* because we can't remember it. There are two ways to define a macro. The first one is using the `\def` command. The `\def` command is followed by the name of your macro, which is then followd by what the macro does within curly brackets. For example:

```
\def\thatword{supercalifragilisticexpialidocious}
```

If we now invoke that macro, it will print *supercalifragilisticexpialidocious* (this was actually printed using the macro, but you can't see that in the PDF).

The other way is to use `\newcommand{...}{...}`, of which the first argument is the name of your macro/command, the second is what it does. For example:

```
\newcommand{\thatword}{supercalifragilisticexpialidocious}
```

One difference between `\def` and `\newcommand{...}{...}` is that, when you define something with `\def` that has already be defined (for example, later on you make a new macro called `\thatword`) you won't get an error. With `\newcommand{...}{...}` you will. If something was defined with `\newcommand{...}{...}` and you want to change it, you need to use `\renewcommand{...}{...}`.

In order to define a new command with arguments, you need to use `\newcommand{...}{...}`. But between the first and the second argument, you put between square brackets how many required arguments your new command will have. Then, in the definition of the new command (i.e. inside the second argument of the `\newcommand{...}{...}`, you reference these required argument by `#1`, `#2`, etc.

So for example, we want a new function that takes two arguments and prints the first in bold and the second in italics. Defining it will look like this:

```
\newcommand{\MyFunction}[2]{\textbf{#1} \textit{#2}}
```

Calling `\MyFunction{John}{Mary}` will now print: **John** *Mary*.

14. Try making a macro or a command yourself.

14 Changing fonts

To change fonts you need—you guessed it—another package: `fontspec`. It takes care of changing fonts on its own, but you have to supply it with the font files first. Click the 'Upload' button in the top-left corner of your screen to select and upload font files to Overleaf.[8] Now we have to tell `fontspec` to go look for these files, by adding `\setmainfont{...}` to the preamble, with the *exact* name of the font file as the obligatory argument. Many font files have specific font sets for **bold**, **italic** and ***bold italic*** formatting. You can pass these on as optional arguments to the `\setmainfont{...}` command. For instance, we can switch to the font *Cambria* by uploading the files `cambriab.ttf` (bold), `cambriai.ttf` (italics), `cambriaz.ttf` (bold italics) and `cambria.ttc` (normal text) to Overleaf and adding the following lines to our preamble:

```
\usepackage{fontspec}
\setmainfont[
    BoldFont={cambriab.ttf},
    ItalicFont={cambriai.ttf},
    BoldItalicFont={cambriaz.ttf}
    ]{cambria.ttc}
```

15 Interpreting errors

When compiling your document, several errors may occur. However, if an error is not very serious, LaTeX will try to compile your document nevertheless. Overleaf tells you exactly how many error there are: the button next to the **Recompile** button is the log—the number on that is the amount of errors. If it's red, there are errors, if it's yellow, there are only warnings. A good tip: you can click on an error to go where it goes wrong in your source code.

But what do these error messages actually mean? It depends. Reading the log really helps. But let's have a look at the most common errors: **Undefined control sequence** and **Overfull \hbox**.

The first means that you are using a command or macro somewhere that LaTeX does not know. It may be a spelling error, but it could also be that you need a specific package for something that you haven't loaded yet.

As for the second—in this document there are a few lines that don't fit, they are too long, for example on the top of this page. Those are "Overfull \hbox"es. `\hbox` stands for horizontal box. They occur very often; before submitting

[8] In Windows, most of these can be found in `C:\Windows \Fonts`.

anything, do check your error log to see if there are any. They are often caused by a word that LaTeX does not know how to hyphenate. If you haven't loaded a language package such as `babel` (which we have, for `biblatex`), it might help to do so. Otherwise, you can tell LaTeX yourself how to hyphenate it.

If it is a word that only occurs once or twice, you can define places where LaTeX is allowed to break a word with `\-` in the text. Otherwise you can, at the top of your document define it once and for all with `\hyphenation{...}`. For example:

```
\hyphenation{hy\-phen\-a\-tion}
```

This will teach LaTeX how it should hyphenate *hyphenation*.

If this doesn't solve your problem, you can try moving words around so that it fits. Otherwise you can (but this is some higher LaTeX skill and is generally regarded a bad move) adjust the **tolerance**. This tolerance sets the tolerance for white space between words: the higher the tolerance, the bigger the white spaces can be. If the white space can be higher, LaTeX will have more freedom to move words around in order to have everything fit nicely in an `\hbox`. For example:

```
\tolerance500
```

sets the tolerance to 500.

If there are errors that you don't understand, do check https://www.overleaf.com/learn/latex/Errors. In general, I really recommend you reading Overleaf's tutorial there, it is very comprehensive.

That's it for today.

If there is something you can't seem to fix or that you want but you don't know how to do it, ask Google. Usually the first hit will the website called StackExchange. I guarantee you, you will find your answer there, unless you have a very obscure problem...

In any way, if you have any questions, do send me an email, I am always happy to help when it comes to LaTeX! And remember, the answer to "Can I do this?" is always yes!