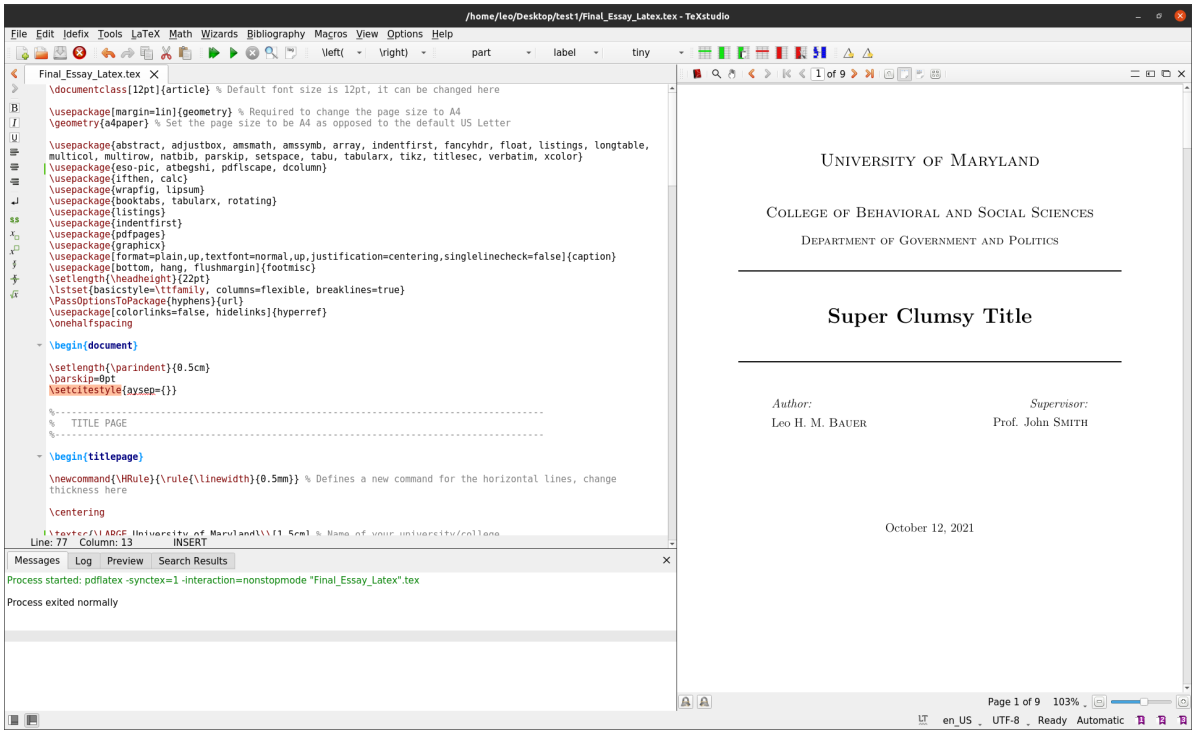
LaTeX Tutorial

Using LaTeX for Scientific Publishing

In this tutorial, we will learn how to create scientific documents using LaTeX. LaTeX is a software system to produce high-quality documents, including articles, books and presentation slides. In contrast to word processors such as MS Word or LibreOffice, LaTeX works with plain text which is compiled into PDF documents.

The software you will be working with to produce LaTeX documents thus looks similar to the picture below. To the left there is a window in which you can edit the document elements, while the compiled PDF is displayed on the right.



LaTeX is used widely in the scientific community, due to its advantages in displaying mathematical notation, precise formatting and easy citations. Precise formatting may be its biggest advantage over classic word processors. The latter do a lot of things automatically when it comes to formatting and moving around tables and figures, which can be tedious especially when editing long documents. Tables may move to the wrong place while writing, formatting decisions may not apply to the whole document and so on. The worst thing is that often there is no way to find out exactly why things are not working as you'd like and thus no clear way to change them. LaTeX in turn leaves most formatting decisions to you, with the result that they work across the document when correctly specified. The big disadvantage is that it takes time to familiarize yourself with the LaTeX environment and set up working templates for, e.g., assignments, conference papers and presentation slides. Yet, once you have these up and running it is easier to produce such documents in LaTeX than in classic word processors.

In sum, LaTeX saves time when working on long documents such as your thesis and when you have working templates. It also has a great look and is convenient when using citations. It is also free (at least the local installations) and open source!

In the present tutorial, we will cover the initial setup, focusing on an easy to use online solution, before going over the basic LaTeX document structure and specifics on text formatting, figures, tables, math notation and citations. We will conclude with a quick look at Beamer, a LaTeX document class that lets you produce presentation slides.

Setup

To work with LaTeX, we need to set up an editor in which to edit documents and an interpreter which supplies the code and packages necessary to compile the final PDF document. You can think about this as the difference between RStudio (editing code) and R itself (the language).

There are two ways in which to work with LaTeX. There are offline solutions so that you can work on documents even if you don't have internet. These may be a bit more tedious to install but generally work quite fast, or at least as fast as your computer allows. If you'd like, you can check **here**, **here** and **here** to install LaTeX locally on Windows and **here** to install it on Mac. Let me know if you have a Linux distribution.

To make things easy, for this tutorial we will use Overleaf, an online LaTeX editor for which no local installation is needed. Go ahead and create an account **here**, by clicking on "Register". Although it is easy to use, compiling longer documents can take quite long in Overleaf or will even time out due to caps on computing resources. For more computing power and for collaborating with others on a document you will have to purchase a subscription. Yet, for the present tutorial Overleaf will be working just fine.

Once you have registered and are on the main page, click the green "New Project" button on the top left and select "Blank Project." Now you will see a minimal working example of a LaTeX document. You can edit text in the window to the left. Click the green "Recompile"

button at the top center of the window to compile what you have written and see the PDF output in the window to the right. Make sure to recompile regularly! While online LaTeX editors may autosave, when working locally this is the only way to save your document.

Basic Document Structure and Workflow

Let's start with the basics. A LaTeX document consists of two parts. First, there is the preamble, in which we declare the document class (for articles, books, CVs, presentations), load packages that allow us to do certain things (add figures, tables, etc.), add formatting options that apply to the whole document (i.e., globally), add a bibliography from which we'll cite and, e.g., create a title page. In the code box below, the preamble is the part until the `\begin{document}` command.

The preamble is followed by the main document, wrapped inside `\begin{document}` and `\end{document}`. You put all text you want to write in here as well as all figures, tables and all else. Importantly, `\end{document}` closes the document, so there cannot be any text after it.

You can add some plain text to the document and click compile to test it. If you want to modify the text in any way (font size, bold, italics, etc.) you will have to wrap the text into a command such as `\textbf{}`, which is the command to make the text bold. For larger things such as tables and figures, you need to use environments that start with `\begin{}` and end with `\end{}`. You can find a cheat sheet with the most common commands **here**.

Throughout the document, you can insert comments that will not be printed when you hit compile using the "%" symbol. This is similar to R, where this is done with the "#" symbol.

```
\documentclass{article}

\begin{document}

Hello!

\end{document}
```

Creating Scientific Documents

Creating scientific documents requires more than simple writing. Specifically, we often need section headers and numbering or wish to display plots, tables or mathematical notations for readers. This can be achieved by commands and by using environments.

Formatting and Structuring Text

Basic commands to format text are `\textbf{}` for bold font and `\textit{}` for italics. We specify the document-level font size by inserting, e.g., `12pt` as an option into the `\documentclass{}` command. This would then look like `\documentclass[12pt]{article}`. To use packages that aid us in formatting, we use the `\usepackage{}` command and specify the package name inside the brackets. Again, we can further specify options using square brackets in front of the curly brackets. For example, we can use English-language spell-checking by inserting the `\usepackage[english]{babel}` command into the preamble.

To structure text into sections, subsections and so on, we can use the `\section{}`, `\subsection{}` and `\subsubsection{}` commands. Simply write the section header into the brackets. These commands will produce numbering that is automatically updated. To not display numbered sections insert an asterisk, such as `\section*{}`. After the section command, you can simply write text until you insert the next section command.

Adding Figures, Tables and Math Notation

Figures

To add plots, tables and math notation we make use of environments. These take the form of a `\begin{}` and an `\end{}` command. Again, inside the bracket we specify the environment we want to use. For any image (plots etc.) this is `figure`, for tables `table` and for math notation `equation`. Then, inside the environment we can insert commands to create figures, tables and math notation. A fully specified figure environment looks like what is in the box below.

```
\begin{figure}[t!]
    \includegraphics[scale=\textwidth]{example.pdf}
    \caption{This is an example plot.}\label{f1}
\end{figure}
```

In the present example, we add the option `[t!]` to the `\begin{figure}` command in order to have it always fixed to the top of the page. If left unspecified, LaTeX will choose the position on the page automatically. The `\includegraphics{}` command then loads the plot, in this case another PDF. Image files such as JPEGs or PNGs work fine as well. I'd recommend to always produce PDFs when plotting in R, STATA or Python, because as PDFs are vector graphics they always look clear, no matter how much you zoom into the document. Make sure to correctly write the file name and put it into the folder in which the LaTeX document is. With the `[scale=\textwidth]` option we set the plot's maximum width to that of the text. Next, we include a caption with `\caption{}` to add a description of the plot and finally a label with `\label{}` so that we can cross-reference the figure in the text. This works with the `\ref{}` command. For example, you'd write `As is evident in Figure \ref{f1}...` and

in the PDF version of the document the number becomes clickable and redirects the reader to the page on which the figure is located. Numbers of figures in the text update automatically.

Tables

For tables, the `\caption{}` and `\label{}` commands work as well. Yet, structuring tables itself can be a bit overwhelming as the code looks quite confusing at first. At best, you simply take a regression model object in R and use an R package to produce you the LaTeX code for a neat regression table. To do this I'd recommend the `modelsummary` package. See the R code chunk below for an example how to use it.

```
modelsummary(m1, title="Logit Model",
             stars = c('*' = .05, '**' = .01, '***' = .001),
             output = "latex")
```

Here, we first specify the regression object name (m1) before giving the table a title, specifying the meaning of the significance stars and then give LaTeX as the output. Once executed, the LaTeX code for the regression table will be printed in the R console, from which you can simply copy and paste it into your LaTeX document.

If you want to create a table from scratch, I recommend a tool like the **Tables Generator**. There you can specify the general scope of the table, including number of rows and columns as well as visible borders and more. Again, the website produces LaTeX code for your table to copy-paste into your document. A basic table looks like in the box below.

```
\begin{table}[t!]
\caption{This is an example table.}\label{t1}
\begin{tabular}{llll}
 &  &  &  \\
 &  &  &  \\
 &  &  &
\end{tabular}
\end{table}
```

After opening the general environment, we see the `[t!]` option, a caption and a label, just as in the figure. Next, we need to open another environment for the table itself, which is `\begin{tabular}`. Make sure to close this one as well with the `\end{tabular}` command, as seen in the code. In the second set of curly brackets we see four l's. This means that all text in the four columns is left-aligned. We can also center it or right-align it. Then these would be four c's or r's, or a mix of all of them. Next come the actual cells of the table. These are delimited by the & signs while the double backslashes denote the line breaks. AS it is, the table is empty. You can simply fill the cells of the table by writing in the empty spaces between

the & signs. As the & sign is used in tables, you cannot use it in-text by simply writing &. Instead, write `\&` if you want to use it in-text.

Math Notation

Math notation is fairly straightforward. After again creating the necessary environment, we can start writing equations. Check the cheat sheet linked above for the most common commands. A code example is provided below.

```
\begin{equation}\label{e1}
4 \times 4 = 16
\end{equation}
```

A short equation like this one might even be better put in the text. You can put any math in-text by wrapping all the math into two $ signs, such as `$\Delta$` for the mathematical symbol for the Greek letter delta. Finally, to capture an equation from a PDF or other document and transform it into LaTeX code to use in your document, I recommend using **this** free tool.

How to Cite

Citations are maybe the most important aspect of a scientific document. To cite in LaTeX, we need a bibliography with the .bib ending in the folder where your LaTeX document is and the proper commands inside the document. Bib-files store literature in a LaTeX format, including a cite-key which you'll use when doing in-text citations. Below is an example of a bibliography item in the bib-format.

```
@article{bellin2012,
  title = {Reconsidering the Robustness of Authoritarianism},
  author = {Bellin, Eva},
  date = {2012},
  journaltitle = {Comparative Politics},
  volume = {44},
  number = {2},
  pages = {127--149},
  doi = {10.5129/001041512798838021}
}
```

How do we produce such a file with literature in the above format? To be sure, you don't want to create such a file manually. I will show you how to do it using **Zotero**, a popular literature management program. Bib files can also be produced using Mendeley, although I really recommend Zotero because it works best with LaTeX and is not proprietary. To create bib-files for use in LaTeX we also need to download and install a Zotero plugin, called **Better**

BibTeX. After installing, simply right-click on a bibliography folder in Zotero and select "Export Collection." Make sure that "Better BibLaTeX" is selected in the top drop-down menu and you can leave the "Translator Options" untouched. As we're working with Overleaf we will have to upload the bib-file anyways, but if you compile LaTeX documents with a local installation, then make sure to tick the "Keep updated" box in the "Translator Options." This feature makes sure that the file is updated once you add new literature in Zotero. It saves a lot of time as you don't have to create a new bib-file every time you add a new item to your bibliography. But beware: This does only work in Overleaf if you become a paying subscriber. Now, simply hit "OK" and save the bib-file in the folder with your LaTeX document. For our present example save it in the Downloads or a similar folder, from which you can upload it into your Overleaf project.

Once uploaded, we need to add the bib-file to the preamble, which looks like in the code box below.

```
\addbibresource{example.bib}
```

Next, we need to specify where to print the bibliography. Usually this is at the very end of the document. The bibliography updates itself once you cite items in your text. We put the bibliography right at the end of all text, just before the `\end{document}` command.

```
\pagebreak

\printbibliography
```

We insert a page break so that the bibliography starts on a new page and then print it there.

After having added the bibliography, we need to select a citation style. This is also specified in the preamble. In the present example we'll use Chicago author-date, but there are many more styles available. We load the BibLaTeX package and load it with the necessary options in the preamble.

```
\usepackage[authordate,backend=biber]{biblatex-chicago}
```

Now we can cite in-text! Use the `\parencite{}` command to cite with parentheses around author names and `\cite{}` with no parentheses. Start typing the author name in the curly brackets and LaTeX will preview the closest matches for you, from which you can select. If you only want to print the year of the citation, add an asterisk like `\parencite*{}` or `\cite*{}`. For multiple citations in a single set of parentheses simply put multiple citation keys separated by commas in the curly brackets. If you want to cite specific pages in a work, simply add square brackets before the curly brackets with the page numbers inside, such as `\parencite[13--14]{}`. Once you compile the LaTeX document, the citation will show in the text while the full bibliography item will be printed in the bibliography at the end

of the document. All in-text citations are cross-referenced, so that the reader is shown the bibliography page with the item when clicking on the in-text citation in the PDF.

The Beamer Class

Creating presentation slides for workshops and conferences is also important. LaTeX has the Beamer class for this task. In principle the document is structured similarly to a LaTeX article document, meaning that there is a preamble, a main part of the document and environments in which you create slides. Once you know how to create LaTeX text documents, it is fairly straightforward for you to learn how to create Beamer presentation slides. We will review an example of how to create a simple presentation in-class.

LaTeX Templates for You

As part of the GVPT Graduate Methods Workshop series, there are templates available for LaTeX documents. You can find templates for a class assignment, a presentation, a CV, an article and a conference poster **here**.