Your First LaTeX Document!

Your Name[‡][*]

[‡]Your Department, Your University

September 22, 2021

Abstract

This document details our first adventures with LaTeX, otherwise known as a session entitled 'A short Introduction to LaTeX: or LaTeX in 105 minutes'. It's going to be delivered as a combination of a more typical lecture style and a 'live-coding' workshop, with short pauses for you to try and implement things as we go. Depending on the number of participants, we might split into groups of two and undertake 'pair-programming' for the exercises. The entire workshop is contained within this .tex document, which can be found at github.com/crahal/latex_in_105, and also acts as a **very short** reference manual/cheat-sheet. There naturally needs to be some slight appreciation of the self-awareness of a document of this type; it's teaching while it's also creating, all-in-one! Most importantly, this is a capacity building exercise for you; not to teach *everything* about LaTeX, but to empower you to learn how to become an expert typesetter in your own time.

Learning Objectives:

- Learn the basic history of, and motivation for using LaTeX.
- Create a basic article document and structure it.
- Include basic tables, figures, and bibliographies into documents.

Some important notes:

- Comments in .tex are shown as %!
- I've tried to add comments to as much as humanly possible in the raw .tex!
– Specifically: every time a new concept/command is identified/utilized.
- Content in red (in the .pdf) can be clicked on!
- Stuff in indented code blocks in the .pdf can be copied and executed!

[*]Acknowledgements/corresponding info here

Contents

1 More Traditional Lecture Content

1.1 Introduction

1.1.1 The Basics of LaTeX

Lets temporarily ignore all of the content in this document up until these lines here. Lets first talk about why we might be interested in using LaTeX:

- LaTeX (\LaTeX) is a document preparation system and document markup language.
- Based on TeX (\TeX), a typesetting system designed by Donald Knuth, released in 1978.

On 30 March, 1977, the diary of Knuth (a computer scientist at Stanford University), recorded the following note to express his dissatisfaction with the quality of typeset proofs he'd just received for volume 2 of his book series The Art of Computer Programming:

"Galley proofs for vol. 2 finally arrive, they look awful... (typographically). I decide I have to solve the problem myself."

TeX is, specifically, a piece of typesetting software that users can control by providing it with instructions written in a special programming language. LaTeX is not the name of any one particular program:

- It refers to encoding or tagging conventions used.
- Technically, LaTeX is a very large collection of complex and sophisticated TeX macros designed to help you typeset books, journal papers and so forth.
- It originally started out primarily the tool of mathematicians and scientists...
- But it would be a misnomer to think it was only for quantitative disciplines.

LaTeX was designed to be extensible: if you visit the Comprehensive TeX Archive Network you can choose from hundreds, if not thousands, of macro packages that have been written and contributed by users worldwide.

- But wider use has lead to the development of a very mature 'programming' language
- Prevalent in all of academia, but more-so in technical subjects.
- Popular for its ability to elegantly handle equations, as we'll see.
- One of the main features of LaTeX is the ability to write a range of complex characters.
- For example, maybe we want to write Arabic?

السَلامُ عَليكُم ورَحمَةُ الله وبَرَكاته

- The aim of today will be to get you towards being able to produce something like this document.
- However! We will not be discussing the technicalities of the compilation process or production of `.gz`, `.log` or `.aux` files.
- Note, importantly, that we wont be talking about how to combine LaTeX and R for report generation (e.g. `Sweave`).

1.1.2 Compared to MS Word:

☹ Disadvantages of LaTeX, compared to a WYSIWYG[1] program such as MS Word:

1. You are unable to see the results instantly.
2. There is a learning curve as in any code language regarding the vocabulary of basic commands.
3. It can often be difficult to generate a specific ‘look’ or ‘feel’.
4. The debugging process can be frustrating, to say the least!

☺ However, as a WYSIWYM language, the benefits are enormous:

1. Layouts, fonts, tables are consistent throughout.
2. Formula are especially easy to typeset beautifully.
3. Indices, footnotes and other references are easily generated.
4. Your documents will have a correct systematic structure.[2]
5. It is **FREE!**
6. Cross-platform: Any .tex file can be compiled through any compiler on any operating system.

1.1.3 Distributions of LaTeX

There are lots of different LaTeX distributions:

1. TeX Live: Many developers, often the distribution of choice on all operating systems.
2. teTeX: A defunct TeX distribution for Unix.
3. fpTeX: A defunct TeX distribution for Windows.
4. emTeX: A defunct TeX distribution for MS-DOS.

[1]Note! You can click this word! Created using the package **hyperref** package.

[2]For example, open these slides in Adobe Reader or many other *.pdf* viewers to see the section headings in the left hand side

5. MiKTeX: An actively maintained TeX distribution primarily for Windows, but also recently ported to Linux and macOS.

6. proTeXt: MiKTeX for begginners (+an editor and ghostscript).

7. MacTeX: same as TeX Live, only for Mac OS X (with better compatability).

The main two choices of distribution are therefore usually between TeX Live and MikTeX. In recent versions, the differences between MiKTeX and TeX Live have narrowed. Package coverage between the two is similar, as is the ability to do on-line/'on-the-fly' updates. MikTeX used to be preferred for Windows and is maintained by Christian Schenk (just one person). TeX Live is maintained by the TeX User Group and pulls directly from The Comprehensive TeX Archive Network.

1.1.4 Different TEX based engines

Each new TEX engine is given its own name to distinguish it from Knuth's original software: hence you now have pdfTeX, XeTeX and LuaTeX. Those three TEX engines are not 100% wholly compatible with each other and it is quite possible to prepare input that can be processed with one engine that also fails to work with others, simply because a particular engine may support primitive commands that the others do not. But all is not lost: enter the world of TEX macros!

So, if someone says they're using LATEX, then this is incomplete information: what engine are they using? What they really mean is that they are using the LATEX macro package with a particular engine, *usually* pdfTeX but maybe XeTeX (for multilingual work or specific font-types) or LuaTeX which incorporates the Lua scripting language (perhaps for advanced customized document production). Frequently you will see terms such as pdfLaTeX, XeLaTeX or LuaLaTeX: but these are not actually the names of TEX engines (i.e. note the difference between XeTeX and XeLaTeX), all they signify is which TEX engine is being used to run the LATEX macro collection, i.e.:

- pdfLaTeX means using the LaTeX macro package with the pdfTeX engine
- XeLaTeX means using the LaTeX macro package with the XeTeX engine
- LuaLaTeX means using the LaTeX macro package with the LuaTeX engine

Today, I am using pdfLaTeX: preparing my typeset document using the LATEX macro package and processing it with the pdfTeX engine. Key features of pdfTeX, XeTeX and LuaTeX

- pdfTeX: ability to output directly to PDF, saving users from having to convert TeX's native DVI format to PostScript and converting that to PDF via GhostScript. pdfTeX also introduced refinements such as margin kerning (character protrusion).
- XeTeX: ability to directly read/input TeX files saved or created in UTF-8, added handling of multilingual typesetting. It also enabled very easy and convenient use of OpenType fonts and later versions added OpenType-based typesetting of mathematics.

- LuaTeX: Arguably the most powerful and versatile of all the TeX engines, derived from pdfTeX (in addition to many other sources/libraries), allowing the Lua scripting language very sophisticated control of the TeX engine. It also supports UTF-8 text encoding, OpenType-based mathematical typesetting and very advanced use of OpenType fonts for text typesetting, although the mechanism is different to XeTeX. LuaTeX also integrates the MetaPost graphics language. LuaTeX is ideal for advanced or customized document engineering: a powerful feature is LuaTeX's extensibility through "plugins" written in C/C++ and loaded as a .DLL (Windows) or .so (on Linux).

1.2 Additional Learning Resources

As discussed at the start of the class, this is a workshop designed to help you become familiar with the general paradigms of LaTeX, rather than becoming an absolute master of all things TeX (which can take decades, if not longer). Therefore, the following additional learning resources might be useful for you as you continue your journey after this class:

- LearnLaTeX.org is a *fantastic* set of 16 'lessons'.
- A First LaTeX Document is very similar in style to this workshop today: it takes you through writing a small document with text and math for the first time.
- Getting Started with LaTeX is a primer with a focus on mathematical notation.
- The Not So Short Introduction to LaTeX is what this course is named after, and is naturally a quintessential resource!
- LaTeX Cheat Sheet: because cheat sheats are cool.

1.3 Installing TeX Live

For the purpose of this class, lets all try and install TeX Live 'over the Internet' from here:

`http://www.tug.org/texlive/acquire-netinstall.html`

by downloading the install-tl-windows.exe file, and clicking through the installation process. Note that TeX Live (since version 2009) and MiKTeX (since version 2.8) both include TeXworks for MS Windows, which is the editor that we're programming in today. If you're running Debian based distributions, the command will be akin to

```
apt-get-install texlive
```

If you want to install on macOS, we'd recommend the MacTeX-2021 Distribution. I'll give you a few minutes to try and get set up there. If you can't get it working for whatever reason, no problem! You can use the *really* handy tool at https://texlive.net/run for the remainder of the class. Once you've got set up (either with TeXworks or the online app), try this Minimum Working Example (MWE):

Our First LaTeX document!

```
\documentclass[article]
\begin{document}
Hello World
\end{document}
```

Have a go at this yourselves, and we'll reconvene in a few minutes, and then move onto more of the live coding section!

2 More Hands On Style LaTeX

2.1 Preamble

As you've seen in this specific .tex file, the top of the file (before `\begin{document}`) loads in packages. This staging area of the file is called the 'preamble'. Elsewhere in the document, you've seen links to websites through the `\href{}` command. What happens if we try and use a `\href{}` without first loading in the appropriate package first? That's right! An error message! Now try it again after first loading in the `\hyperref` package. Do note that this is just one of an *unbelievably large* number of packages available on CTAN!

Adding a package to the preamble

```
\documentclass[a4paper, 12pt]{article}
\usepackage{hyperref}
\begin{document}
\href{youtube.com/watch?v=QH2-TGUlwu4}{The Best Thing on YouTube}
\end{document}
```

Hint: You can copy and paste this text in these verbatim environments into your IDE!
Quiz: Can you figure out how to color the clickable link?

2.2 Title Page

With almost every `article` class document, you'll want a titlepage. This document has one, too! How about you take the frontmatter/titlepage content from this document, and then create a MWE with an author, title and affiliation? You can put anything you like into these fields in this example; be creative! Perhaps you're a super-hero, or your favourite animated cartoon character today?

Our First Title Page

```
\documentclass[a4paper, 12pt]{article}
\usepackage{authblk}
\begin{document}
\title{Your First \LaTeX{} Document!}
\date{\today}
\author{Your Name$^{\ddagger}$}
\affil{$^{\ddagger}$Your Department, Your University}
\maketitle
\end{document}
```

We can also build on this titlepage even further to make it more complete, such as including an abstract, or corresponding information. We can also take this as an excuse to introduce the `lipsum` package, which allows us easy access to Lorem Ipsum: a placeholder text commonly used to demonstrate the visual form of a document or a typeface without relying on meaningful content (i.e. it uses a combination of characters which look like real text). Note in the below how we've stacked two packages into one `\usepackage{}` command; pretty neat, huh!

Our First Title Page

```
\documentclass[a4paper, 12pt]{article}
\usepackage{authblk, lipsum}
\begin{document}
\title{Your First \LaTeX{} Document!}
\date{\today}
\author{Your Name$^{\ddagger}$\thanks{To the Academy}}
\affil{$^{\ddagger}$Your Department, Your University}
\abstract{\lipsum[1]}
\maketitle
\end{document}
```

Lets take a couple of minutes to put together our own titlepages. Perhaps you want to share yours?

2.3 Text and Characters

2.3.1 Commenting Out

Right! Now we've got our titlepage set up, we can get down to the fun stuff; writing up our research! In the verbatim blocks that follow, I'm going to jettison our titlepage, if that's permissible! First of all, lets talk about how we 'comment out'; as with almost all programming languages, this is a way of embedding text (in the form of notes, guidance, or information for our future selves or collaborators) which is not executed by the TeX engine nor compiled into our document. With TeX, this looks like a % sign. We're now going to see two types of 'comments'; one inline, and one out of line. Copy it into your editor and compile it, changing some of the 'inline' comments as appropriate:

Commenting Out

```
\documentclass[a4paper, 12pt]{article}
\begin{document}
% This is a comment outside of a line, and below is ‘inline’:
While it might not look like it, this is an %irrelevant
%irresistible
%irrefutable
%irredeemable
example of how to comment something out.
\end{document}
```

2.3.2 Text Styling

Now that we've learnt how to comment things out, we can talk about how to style our text. The five main types of styling are 1.) bold, 2.) italics, 3.) ‘emphasis’, 4.) underline, and 5.) typewriter font. Note that while `\emph` and `\textit` might look similar when rendered, they're subtly different: you shouldnt use `\emph` when you simply want something in italics, because other packages or styles might modify it to give a *slightly* different emphasis on a word (e.g. underline), and we'll see an example of when it's embedded later in this section.

Italicized text: Some of the greatest discoveries in science were made by *accident*.
Boldface text: Some of the **greatest** discoveries in science were made by accident.
Underlined text: Some of the greatest discoveries in <u>science</u> were made by accident.
Typewriter text: `Some` of the greatest discoveries in science were made by accident.

Class Quiz: Can you combine all of these stylings into one sentence?

A few final utilities when it comes to styling text, although not the text itself but the actual structure of it. lines. The first is to break lines/create new lines: there are actually four common ways to do this!

1. By leaving an empty line in your text editor
2. \\ (two backslashes)
3. By using `\newline`
4. And by using `\hfill \break`

You might also be interested in clearing the page (with `\clearpage`), or creating a new page (with `\newpage`). These two commands are similar, but have subtle differences in how they treat stacked floating elements (we'll see more of them later). Other commands which might be of interest are adding in horizontal (`\hfill{xin}`) or vertical (`\vfill{xin}`) space, where `x` is a floating point measure of inches to pad. We can also talk about the alternate option of `\quad`, which is technically a LaTeX macro for:

```
\def\quad{\hskip1em\relax}
```

but that's a more advanced discursion for another day/self-study, perhaps!

2.3.3 Colors

As we've already seen in other parts of this document, its naturally possible to colour in parts of text. This is done with the simple `\color{}` command. The simplest manner to use colours in your LaTeX document is by importing the `xcolor` package. Both packages provide a common set of commands for colour manipulation, but the latter is more flexible and supports a larger number of colour models so is the recommended approach. For example:

This text is in red!
And this text is in blue!
This text is black, but the line below it is purple:

We havent seen the ability to draw ruled line across the page so far, so lets quickly note that this is done with `\rule{\linewidth}{0.5mm}`. We can also use `\colorbox{}` to change specific backgrounds of things, and if we want a different framecolor on our colorbox, we can use `\fcolorbox{}`. For example;

I'm text inside a colorbox! Nice to meet you!
My text can *also* be colored! Nice to meet you!
And so can my frame!

Lets quickly note that the last colour (ForestGreen) is not in the standard `xcolor` package; we had to augment our loading of `xcolor` with an argument: `[dvipsnames]`. Two final things to note; the middle red background colour is *slightly* different; red vs. Red. We can also define or redefine colours with `\definecolor{black}{rgb}{0.1,0,1,0.1}` (which would redefine black to be slightly lighter; try it out!), but we haven't done so in this document.

Class Quiz: Can you take the *emphasized*/**bold**/*italic*/`texttt` string from above, and colour it like this (i.e. with a green frame, red background, with blue text)?

2.4 Lists

There are two types of list in LaTeX, both of which we've already seen in this document in one way or another. The first is an *un-ordered* list, created using the `\begin{itemize}` command, and the second is an *ordered* list, created using the `\begin{enumerate}` command. Lets make a list of things that we need to do today:

```
\begin{itemize}
\item Feed the cat.
\item[!] Pay council tax.
\item[$\blacksquare] Water the plants.
\end{itemize}
```

.
Note here that I've actually styled the bullet points in some specific way! There is *a lot* of customizability to do with lists like this. Note also that we can nest lists within lists, too, up until a certain level! Actually, lets set this as a slightly more difficult class quiz ...

Class Quiz: How about you have a go at making an **enumerated** list of your three favourite genres of movie, while you nest within them an **itemize** (i.e. unordered) list of three of your films in that genre (an impossible question, I know)...? My answer is below, but note that I've actually `input` this LaTeXin from another file, just to be sure that you can't cheat!

1. Anime!
   - My Neighbour Totoro
   - When Marine Was Here
   - Your Name
2. Heist
   - Oceon's 11
   - The Town
   - The Dark Knight
3. Romantic Commedies
   - As Good as It Gets
   - 10 Things I hate About You
   - Bridget Jones's Diary

Following this, lets `\newpage` because we're going to move on to a whole different set of more technical issues next.

2.5 The Beauty of Mathematics in LaTeX

2.5.1 Loading Math Mode

As discussed above, one of the main motivations for using LaTeX is just how elegantly it can typeset equations. In particular, you will likely want to load in two extension packages published by the American Mathematical Society: `amssymb` and `amsmath`. Thus, our preamble will need to include:

```
\documentclass{article}
\usepackage{amssymb,amsmath}
```

The `amssymb` package might be omissible for documents whose math symbol usage is relatively modest. If we want to enter math mode and use 'inline' formulas, we can use a dollar sign ($) to enter and exit, such as this: $a = b+c$. To utilize an unnumbered equation in 'display mode', we use something such as:

```
\begin{equation*}
a=b+c
\end{equation}
```

Or, if we want our equations to be automatically numbered, we can remove the asterisk (*). A simple single line equation as an example can be seen below:

$$a^2 = b^2 + c^2 \tag{1}$$

How about a much more elaborate example? Lets now see for example some symbols (such as ∞, π), lim, $\sum$ and $\frac{frac}{tion}$ (and also more examples of superscript and subscripts):

$$\lim_{n \to \infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6} \tag{2}$$

We can also split equations over multiple lines:

$$\begin{split} a &= b + c - d \\ &\quad + e - f \\ &= g + h \\ &= i \end{split} \tag{3}$$

Note how we've labelled this above equation, and we can cite it back here: 3. Note: this is the first time we've seen `\label{}` and `\ref{}`, but we'll also be labelling other things further down the document: sections, figures, tables, etc.

2.5.2 Styling Math Mode Content

Lets go back a step and talk about styling our math mode content. For example, we can add a $\hat{hat}$ on things with the `\hat{}` command (e.g. for estimated values), or a $\bar{bar}$ over things with the `\bar{}` command (e.g. for mean values). We can also include all of our favourite greek characters: Γ, Δ, Λ, Φ, Π, Ψ, Σ, Θ, Υ, Ξ, Ω, α, β, γ, δ, ϵ, ζ, η, θ, ι, κ, λ, μ, ν, ξ, π, ρ, σ, τ, υ, ϕ, χ, ψ, ω, ε,φ, ϖ, ϱ, ς, ϑ. We can also use $\mathbb{B}$lackboard Bold letters with the `\mathbb{}` command, just like we can also use $\mathcal{C}$aligraphic letters using the `\mathcal{}` command. We can also use some other fancy mathematical symbols, like $\clubsuit$, $\nexists$, $\varnothing$, etc.

2.5.3 Matrices

There are *so many* different types of matrix environments. Lets look at three of the main ones, in the form of `pmatrix`, `bmatrix`, and `matrix`.

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{matrix} = \begin{bmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ p_{21} & p_{22} & \ldots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{bmatrix} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix} \tag{4}$$

We can also use the \begin{smallmatrix} and \end{smallmatrix} commands to get a matrix inline, like this: $\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}$.

2.5.4 Selected Other Various Mathematical Concepts

The command \sqrt produces a square root. To specify an explicit radix, give it as an optional argument, e.g.:

$$\sqrt{\frac{n}{n-1}S}$$
$$\sqrt[3]{2}$$

Other cool things to be aware of before we move on include 'continued fractions' (i.e. the \cfrac{}{} command;

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}}$$

and the use of \{} style non-italicised font within math mode itself. For example, try out this MWE yourselves:

Text in Math Mode

```
\documentclass{article}
\usepackage{amssymb,amsmath}
\begin{document}
\begin{equation*}
f_{[x_{i-1},x_i]} \text{ is monotonic,}
\quad i = 1,\dots,c+1
\end{equation*}
\end{document}
```

2.6 Figures

2.6.1 A Simple Figure

From time to time, it's necessary to add pictures to your documents. LaTeX will index figures automatically and tag them with successive numbers when using the `figure` environment and the `graphicx` package. Lets see our first example of figures with this picture of a cat with a monocle. I have no idea where this picture comes from, but it's – for one reason or another – always been part of my LaTeX material.

There are a few things going on with this example. First of all, note the use of \begin{figure} to open our figure environment (which takes care of the numbering and positioning of the image within the document), and the use of \end{figure} to close it. We can also see the picture is centered through the use of \begin{center} and \end{center}. We've also

Figure 1: Cat with monocle

set a caption with `\caption{Cat with Monocole}`, a label as above, and then actually included the cat with the `\includegraphics` command (and scaled accordingly) with a relative path to the file.

2.6.2 Figure Positioning

At some point, you will notice that the figure doesn't necessarily show up in the exact place as you put your code in the .tex file. If your document contains a lot of text, it's possible that LaTeX will put the picture on the next page, or any other page where it finds sufficient space. To prevent this behavior, it's necessary to set the float value for the figure environment. To do this – as seen above – we put a value in square brackets something akin to [!h] which means to 'force the figure to stay here'. Possible settings include:

- **h** (here) – same location
- **t** (top) – top of page
- **b** (bottom) – bottom of page
- **p** (page) – on an extra page
- **!** (override) – will force the specified location

2.6.3 The `subfigure` environment

The `subfigure` environment allows you to place multiple images at a certain location next to each other. First you need to add the `subcaption` package to your preamble, and then you need to add multiple `\begin{subfigure}` environments within a figure environment. Take a couple of minutes, and try to download two figures (memes mean morale) and nest them side by side. We'll then reconvene, and I'll talk through the code in `more_cats.tex`. Note: the caption is now *below* the cats.

(a) Cat (b) The same cat

Figure 2: The same cat. Twice.

2.7 Tables

2.7.1 Simple Tabular Examples

Tables are ever so slightly more tricky than figures. Note, that we don't need any initial packages here, and so our verbatim of the simplest possible example is as follows:

My First Table

```
\documentclass{article}
\begin{document}
\begin{tabular}{ c c c }
 cell1 & cell2 & cell3 \\
 cell4 & cell5 & cell6 \\
 cell7 & cell8 & cell9
\end{tabular}
\end{document}
```

The tabular environment is the default LaTeX method to create tables. You must specify a parameter to this environment; here we use {c c c} which tells LaTeX that there are three columns and the text inside each one of them must be centred. Each & is a cell separator and the double-backslash \\ sets the end of this row. We can also use vertical separators to include vertical lines in the table (like this {c | c | c}), and horizontal lines with \hline. We can also use the `array` package to add fixed width cells to the table:

cell1 cell1 cell1	cell2	cell3 cell3
cell4 cell4 cell4	cell5	cell6 cell6
cell7 cell7 cell7	cell8	cell9 cell9

In this example, the *m* sets the content of the cell to the middle, and we have three different ways to specify how wide we would like the cells to be.

2.7.2 Simple Float Table Environments

Positioning a table is easy if they're inside a float table environment, using the combination of [h], [t], [p], [b] and [!] as described above. Similarly, we can also include captions and labels just as we've learned about above. How about we see a more advanced table example which includes these, use `\centering` and also row headings, too? Lets also force this table to the bottom of the page, filling in until there using the lipsum command:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae,

Index	Variable 1	Variable 2	Variable 3
1	a	b	c
2	b	c	d
3	e	f	g

Table 1: Table to test captions and labels.

felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Great! This looks like a nice table above. Here is the code for that table:

My Second Table

```
\documentclass{article}
\begin{document}
\begin{table}[b!]
\centering
\begin{tabular}{c c c c}
\hline
 Index & Variable 1 & Variable 2 & Variable 3 \\ [0.5ex]
 \hline
 1 & a & b & c \\
 2 & b & c & d \\
 3 & e & f& g \\ [1ex]
 \hline
\end{tabular}
\caption{Table to test captions and labels.}
\label{our_first_table}
\end{table}
\end{document}
```

Class Quiz: How about you make a new table, just to practice? How about you make a table with the subject in the index, how difficult you find it as a second variable, how fun that subject is, and how useful you find it. My example can be found below, with the code input from the supplementary folder just in case you need a hint:

Table 2: Our Favourite Subjects

Subject	Difficulty	Fun	Usefulness
Econometrics	10	10	10
Finance	8	9	10
Macroeconomics	9	7	9
Microeconomics	7	4	1

2.8 References

2.8.1 A Basic Bibliography with `biblatex`

There are three main options in LaTeX: `bibtex`, `natbib` and `biblatex`. We're going to focus on the use of the `biblatex` package to manage and format bibliographies: a modern option which is perhaps a maybe slightly easier and more flexible interface than the other two options. Lets look at an extremely simple MWE in our verbatim environment which has got some new, bibliography-specific commands to generate a citation like this (the coolest paper from the last few years): [1]. How have we generated this? Here's the verbatim MWE:

My First Citation

```
\documentclass{article}
\usepackage{biblatex} %Imports biblatex package
\addbibresource{sample.bib} %Import the bibliography file
\begin{document}
Machine Learning is cool, as shown by \cite{silver2017mastering}.
\printbibliography %Prints bibliography
\end{document}
```

There are a few new and important things going on here. We're loading a package like we might expect, but we're adding a resource called `sample.bib`. Note, here, that `.bib` files are how we contain information about each referenced book, article, and so forth that we want to cite. We'll see an example of a `.bib` file below. Then, the next new command that we see is \`cite`{}, where we call the identifier that we want to cite from the `.bib` file. To print out all of the references that we are citing in the document, we use the `\printbibliography` command. There are lots of options we can pass to `biblatex`, such as:

```
\usepackage[
backend=biber,
style=alphabetic,
sorting=ynt
]{biblatex}
```

which sets the backend engine (e.g. `biber`, the style of the bibliography (alphabetic), and how the entries are sorted (e.g. sorted by year, name and title).

2.8.2 The Bibliography File

Bibliography files must have the standard bibtex syntax. As a quick aside, getting this right takes some practice, and can be a common source of frustration. Fortunately (and it comes as recommended), you can download .bib entries directly form scientific resources such as Google Scholar. Lets see an example file below which cites [1]:

```
@article{silver2017mastering,
  title={Mastering the game of go without human knowledge},
  author={Silver, David and Schrittwieser, Julian and Simonyan,
Karen and Antonoglou, Ioannis and Huang, Aja and Guez, Arthur
and Hubert, Thomas and Baker, Lucas and Lai, Matthew and Bolton,
Adrian and others},
  journal={nature},
  volume={550},
  number={7676},
  pages={354--359},
  year={2017},
  publisher={Nature Publishing Group}
}
```

This file contains records in a special format; for instance, the first bibliographic reference is defined by the following types of information. `@article...`: This is the first line of an entry, telling us the type of entry (in this case an article). `silver2017mastering` is our identifier assigned to this entry; uniquely used to refer to this article within the document. `author` indicates the authors of the article, where several other comma-separated fields can also be added using the same syntax key={value}, (e.g. volume, number, pages, year, etc). Finally, our reference section is shown below:

References

[1] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.