

Adventures in L<sup>A</sup>T<sub>E</sub>X  
or  
I'll be T<sub>E</sub>X

Amanda Charbonneau  
Michigan State University

February 9, 2016

# Contents

<b>I</b>	<b>Introduction</b>	<b>2</b>
<b>1</b>	<b>Before we begin</b>	<b>3</b>
1.1	How to use this tutorial . . . . .	3
1.2	Getting an Overleaf account . . . . .	3
<b>2</b>	<b>Using L<sup>A</sup>T<sub>E</sub>X</b>	<b>3</b>
2.1	Why are we even doing this? . . . . .	3
2.2	A brief (but amusing) history of L <sup>A</sup> T <sub>E</sub> X and T <sub>E</sub> X . . . . .	6
<b>3</b>	<b>Using Overleaf</b>	<b>6</b>
<b>II</b>	<b>Basic L<sup>A</sup>T<sub>E</sub>X</b>	<b>7</b>
<b>4</b>	<b>Setting up a new document</b>	<b>7</b>
4.1	Starting fresh . . . . .	7
4.2	Editing Settings . . . . .	8
4.3	My first words . . . . .	9
4.4	Top Matter . . . . .	11
4.5	Packages . . . . .	11
4.6	Document Classes . . . . .	14
4.6.1	Document Class Options . . . . .	16
4.7	Organization . . . . .	18
<b>5</b>	<b>Non-paragraphs</b>	<b>20</b>
5.1	Lists . . . . .	20
5.2	Tables . . . . .	21
5.2.1	Better Tables . . . . .	23
5.3	Equations . . . . .	24
5.3.1	Better Equations . . . . .	26
5.4	Images . . . . .	28
5.4.1	Better Images . . . . .	29

<b>6</b>	<b>Referencing</b>	<b>30</b>
6.1	Self Reference . . . . .	30
6.2	Footnotes . . . . .	32
6.3	Bibliographies . . . . .	33
6.3.1	A better bibliography . . . . .	34
6.3.2	An even better bibliography . . . . .	36
<b>7</b>	<b>Collaboration</b>	<b>37</b>
7.1	Sharing . . . . .	37
7.2	Version Control . . . . .	38
7.2.1	Minor versions . . . . .	38
7.2.2	Major Versions . . . . .	38
7.3	Commenting . . . . .	39
7.4	Legit Git . . . . .	40
7.4.1	Overleaf into a clean director . . . . .	40
<b>III</b>	<b>Customization</b>	<b>40</b>
<b>8</b>	<b>Finer Text Control</b>	<b>40</b>
8.1	Fonts . . . . .	41
8.1.1	Sizes . . . . .	41
8.1.2	Families . . . . .	42
8.1.3	Spacing . . . . .	44
8.2	Emphasis . . . . .	47
8.3	Colors . . . . .	47
8.4	Defining your own commands . . . . .	48
8.4.1	Custom colors . . . . .	48
8.4.2	Arbitrary Custom Formatting . . . . .	49
<b>9</b>	<b>Custom Images</b>	<b>50</b>
9.1	Images . . . . .	50
9.2	Graphs . . . . .	51

# Part I

## Introduction

### 1 Before we begin

#### 1.1 How to use this tutorial

This PDF was generated at Overleaf, and is available at [this link](#).  $\text{\LaTeX}$  is plain text, so formatting commands are blended into the text. Text you should type into your editor will appear inside grey boxes, for example:

```
\section{This is a section title}  
Words, words, words.
```

#### 1.2 Getting an Overleaf account

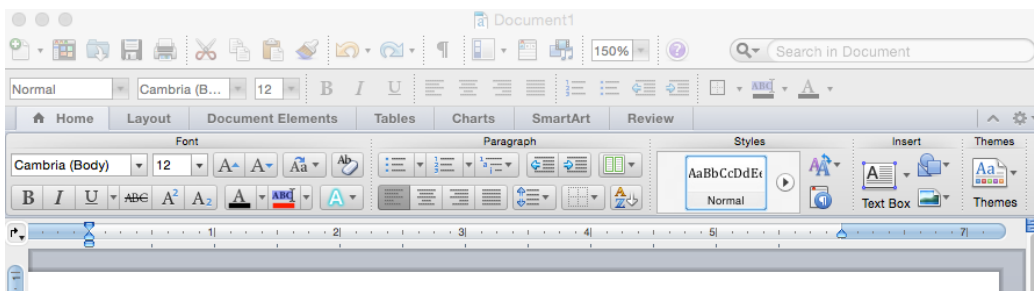
For today's lesson, we're going to use Overleaf and GitHub(**please click those link and set up accounts if you haven't already.**) You'll also need to have Git installed locally.

We're using Overleaf entirely because it's the first  $\text{\LaTeX}$  web client I encountered, so I know it best. After this tutorial, you may also want to check out: ShareLaTeX, Authorea, or whatever the new hotness is that comes up on Google.

### 2 Using $\text{\LaTeX}$

#### 2.1 Why are we even doing this?

Think about Microsoft Word.



When it opens, you have not only a blank page on which to write your thoughts, but also dozens and dozens of toggles and switches and menus. Sometimes we want ultra-precise control over exactly how a bit of our document looks and where it is placed; but usually not. Most times, we just want our document to fit a set of rules:

1. The thesis rules for your university
2. The submission rules for a funding agency
3. The author rules for a journal

With all the buttons available in Word, it *seems* like this should be easy, but often it's not. We've probably all had the experience of trying to format a r sum  or CV in Word, where bolding a word on the second page inexplicably and irrevocably centers the whole first page. However, L<sup>A</sup>T<sub>E</sub>X excels at formatting a document to fit a rule structure. Furthermore, re-submitting that same proposal or manuscript to another agency or journal who has *completely different rules* is easy. In many cases, the switch can be made by editing only a couple lines near the top of your document. Plus, L<sup>A</sup>T<sub>E</sub>X is great at producing equations (it was originally made with math in mind), and many of the commands are easily guessable for native english speakers.

Making documents in L<sup>A</sup>T<sub>E</sub>X is simultaneously much easier and much more difficult than working in more familiar formats like Word or Pages. It's easier, because rather than fighting with invisible characters, indents, numbering, fonts; you just write text. L<sup>A</sup>T<sub>E</sub>X uses obsessively calculated, typographically inspired rules to place your words, images, charts and tables into a pleasing format. It's more difficult, because although you don't need to make the decisions about placement, or keep track of which reference is number 27, you *do* need to identify each bit of your document, so L<sup>A</sup>T<sub>E</sub>X knows what to do with it. The learning curve can be steep, but once you get used to the

system, the benefits far outweigh the occasional troubles. Here's a concise list of the pros and cons, according to the  $\text{\LaTeX}$  wikibook[5]

**Cons:**

1. You don't (usually) see the final version of the document when editing it.
2. You generally need to know the necessary commands for  $\text{\LaTeX}$  markup.
3. It can sometimes be difficult to obtain a certain look for the document.

**Pros:**

1. Document sources can be read with any text editor and understood, unlike the complex binary and XML formats used with WYSIWYG<sup>1</sup> programs.
2. You can concentrate purely on the structure and contents of the document, not get caught up with superficial layout issues.
3. You don't need to manually adjust fonts, text sizes, line heights, or text flow for readability, as  $\text{\LaTeX}$  takes care of them automatically.
4. In  $\text{\LaTeX}$  the document structure is visible to the user, and can be easily copied to another document. In WYSIWYG applications it is often not obvious how a certain formatting was produced, and it might be impossible to copy it directly for use in another document.
5. The layout, fonts, tables and so on are consistent throughout the document.
6. Mathematical formulae can be easily typeset.
7. Indexes, footnotes, citations and references are generated easily.
8. Since the document source is plain text, tables, figures, equations, etc. can be generated programmatically with any language.
9. You are forced to structure your documents correctly.

---

<sup>1</sup>What You See is What You Get, e.g. Word

## 2.2 A brief (but amusing) history of L<sup>A</sup>T<sub>E</sub>X and T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X is essentially a markup language, and is a macro for T<sub>E</sub>X, and similar to other higher level languages in that it provides an easier to use language and expanded capabilities for T<sub>E</sub>X. The 'La' in L<sup>A</sup>T<sub>E</sub>X stands for 'Lamport', specifically Leslie Lamport, who released the original in 1985. The T<sub>E</sub>X is supposed to be an abbreviation of the Latin 'Texnh' (Greek for both "art" and "craft") and is therefore the letters tau, epsilon, and chi. According to its creator Donald Knuth, it should rhyme with 'Bach', however Knuth seems to be the only person who actually does that (see also: gif.)

Knuth created T<sub>E</sub>X because book printers completely changed their technology between the first and second edition of his book *The Art of Computer Programming*, which necessitated an entirely new typesetting... an activity he evidently found so tedious that he preferred to write an entirely new language instead. As this was likely to be a problem that recurred every time the physical technology changed, he wrote T<sub>E</sub>X to be entirely platform independent—it will produce exactly the same output on any printer or computer. Furthermore, it is 100% reproducible, and should produce exactly the same output *forever*. Knuth froze development on T<sub>E</sub>X in 1989 at version 3, and all subsequent changes have contained only bug fixes. Since the freeze, releases have been numbered such that each new version number asymptotically approaches  $\pi$ . The current stable release is 3.14159265. Although the project is in the public domain, and others are allowed to change and improve on them, Knuth controls the original T<sub>E</sub>X and other versions are not allowed to use that name. In fact, Knuth is so devoted to both reproducibility and oddness that he has said that not only will he be in control of all releases while he is alive, but also that upon his death the version number is to be changed to exactly  $\pi$ , and that "From that moment on, all 'bugs' will be permanent 'features'." [3]

## 3 Using Overleaf

Previously, you had to run L<sup>A</sup>T<sub>E</sub>X locally, and it was much like running base R or Python—you would write mostly in a plain text editor, and wait to execute all the code at the end. This sort of system is great once you know a language, but can make debugging or learning a new language pretty challenging. Luckily, services like Overleaf make this process a lot easier. Over-

leaf, to continue the analogy, is like RStudio or iPythonNotebook. It shows your source document and output simultaneously, while also offering useful hints and auto-completion. Since they're online, Overleaf, and other similar services, also offer automatic version control, optional Git integration and lots of easy to use collaboration tools. For instance, you can send your collaborator who is allergic to coding a link to a rich-text version of your Overleaf document, where they can edit and make comments in a Word-like environment.

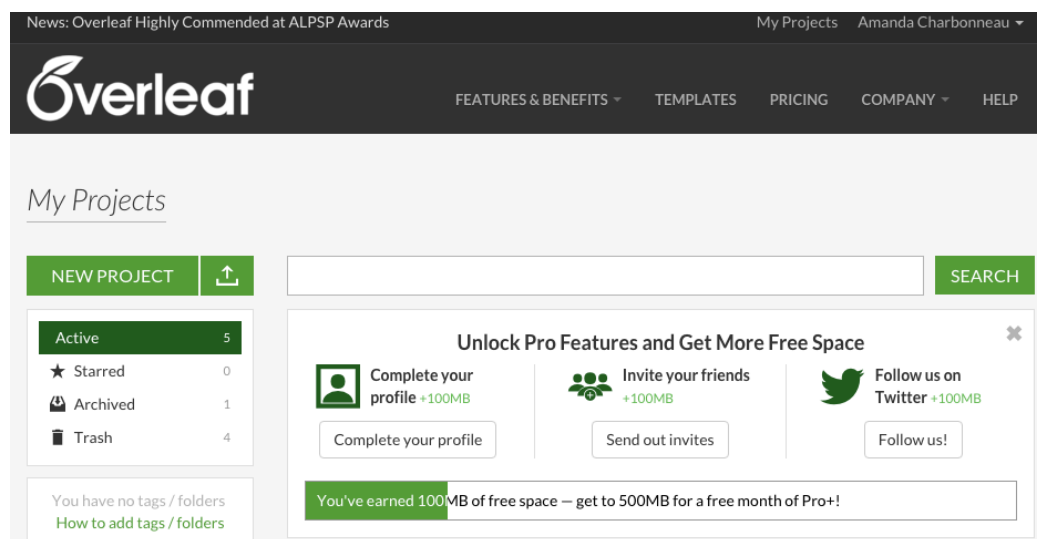
## Part II

# Basic L<sup>A</sup>T<sub>E</sub>X

## 4 Setting up a new document

### 4.1 Starting fresh

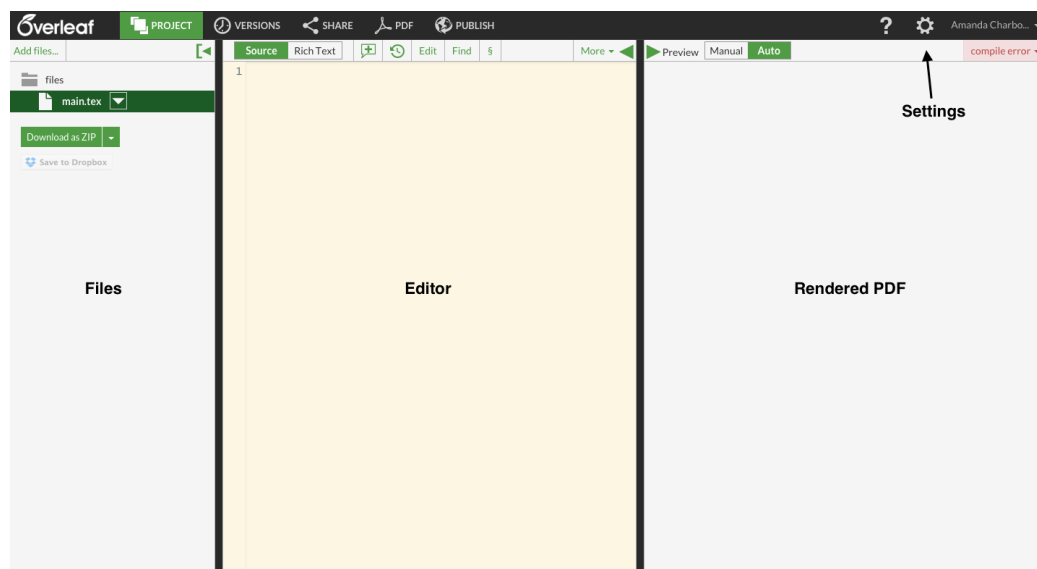
Log into your Overleaf account, and you should see a page that looks like this:



Click on the *NEW PROJECT* button, Overleaf will offer you several pre-made templates to try, but we're going to start from scratch, so select the



first option under *basics*: blank paper. You should get a page that looks like this:



If you can't see the narrow column on the left, it can be toggled on and off by clicking *Project*.

You'll start with just one file: `main.txt`. It's empty (as shown by your editor), and, rather predictably, currently renders nothing.

## 4.2 Editing Settings

The gear icon on the top right allows you to adjust settings for both the specific document that you're working on, and your global environment. The default settings for your project are probably fine, and you shouldn't change them for this tutorial. However, you may find working in Overleaf more comfortable if you change the text editor to one that you use frequently (vim or emacs) rather than the default, and change the text size, background color and spell check language to something you like. I recommend leaving both the auto-close brackets and auto-complete commands boxes checked, as this makes the site *much* easier to use.

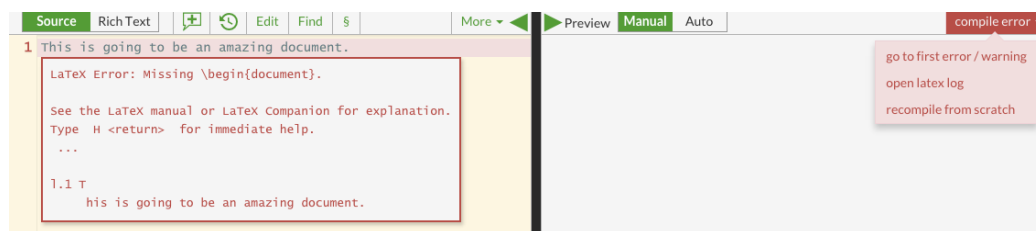
Outside the settings, you can also toggle the **Preview** pane between **Auto** and **Manual**

## 4.3 My first words

In your editor, type some words:

```
This is going to be an amazing document.
```

Once your document has refreshed, you should get a compile error:



This is because unlike a typical document editor,  $\text{\LaTeX}$  doesn't handle naked text well. Before  $\text{\LaTeX}$  can properly format our document, it needs to know several things:

1. What *kind* of document are we making? A book? a PowerPoint? An article? Something else?
2. Where exactly *is* this document? or more specifically, where should compiling begin and end?

First, let's get rid of this compile error. The error box itself gives us a few suggestions, but the easiest way to is to comment out this line, so  $\text{\LaTeX}$  stops trying to read it. We do that by putting a % at the beginning of the line:

```
%This is going to be an amazing document.
```

You'll notice that we do still have a compile error, but now the error is that there's nothing to compile. Now let's address our two formatting points. We want to write an article. We do that by setting the document class, which is specified by: `\documentclass[options]{class}`

```
\documentclass[12pt]{article}  
%This is going to be an amazing document.
```

Setting the document class should almost always be the first non-commented line of your document. Since not everything in our document is going to be text, we also need to tell  $\text{\LaTeX}$  where to start compiling. We do that by

making a document environment, that is, by telling L<sup>A</sup>T<sub>E</sub>X where the document begins and ends. (We'll be setting up lots more environments inside this one later.) `\begin{document}...\end{document}` statement:

```
\documentclass[]{article}
```

```
\begin{document}
```

```
This is going to be an amazing document.
```

```
\end{document}
```

(Note that at this point, we can uncomment our line, and get a good pdf preview.)

Lets add a short second paragraph:

```
\documentclass[]{article}
```

```
\begin{document}
```

```
This is going to be an amazing document.
```

```
It really is!
```

```
\end{document}
```

Like markdown and several other plain text languages, L<sup>A</sup>T<sub>E</sub>X doesn't use single carriage returns in the source file, and is instead interpreting our two sentences as being in the same paragraph. We can fix this in two different ways: By adding a second carriage return

```
This is going to be an amazing document.
```

```
It really is!
```

**OR** by explicitly specifying a line break with `\\`

```
This is going to be an amazing document.\\
```

```
It really is!
```

Notice that these two options differ slightly in their output: Double spacing the source between paragraphs results in the correct rendering, where the

second paragraph is indented. The `\\` just breaks the line without starting a new paragraph. Generally, it's best to use the former in text, and the latter when formatting tables and other special situations.

## 4.4 Top Matter

Most  $\text{\LaTeX}$  commands follow the same format:

```
\command[options]{keyword or text to do command on}
```

and many commands use standard english as the command name, and do exactly as you'd expect. Let's add a few more standard things to our document so we have something to look at:

```
\documentclass[]{article}

\begin{document}

\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle

This is going to be an amazing document.

\end{document}
```

Note that adding a title, author and date doesn't do anything to our pdf until we add `\maketitle`, which signals to  $\text{\LaTeX}$  that we've finished adding 'top matter'...which is the  $\text{\LaTeX}$  equivalent of metadata. Although there is no explicit call to 'top matter', you'll see lots of references to it in documentation of packages.

## 4.5 Packages

There are a few packages you will want to load for pretty much every document you're going to make. The basics are:

Package	Call	Uses
babel	<code>\usepackage[english]{babel}</code>	language specific formatting
amsmath	<code>\usepackage{amsmath}</code>	most common math symbols
inputenc	<code>\usepackage[utf8]{inputenc}</code>	allows common character encodings
graphicx	<code>\usepackage{graphicx}</code>	enhanced graphic support

To load a package, we use the `\usepackage{}` command. We want the package to load *but not be part of the text*, so it has to go in the **Preamble**, that is, after the `\documentclass{}` call, but before the `\begin{}` statement. Packages that don't require options can be put in the same call, but it's usually easier just to do one per line.

Right now, this document is pretty empty, so let's get some text to play with. Rather than typing a bunch of paragraphs, we're also going to use a package to get some lorem ipsum filler:

```
\documentclass[]{article}

\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{lipsum}
\usepackage{float}

\begin{document}

\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle

This is going to be an amazing document.

\end{document}
```

At this point, we have *loaded* the lipsum package, but we haven't used anything in it yet. Just like R or Python, each package will have been written by a different person or group, and will have it's own documentation. Generally, they will also use the `\command[options]{keyword or text to do command on}` format, however Overleaf won't prompt you with possible options, so you'll

have to actually look up information on each non-standard package you use.

The `lipsum` package contains 150 paragraphs of text, and doesn't require a keyword. Just use the desired paragraph number or number range as the options. So to get two paragraphs of text, right after our current sentence:

```
This is going to be an amazing document.  
\lipsum[1-2]  
\end{document}
```

#### Challenge 4.1

Use `lipsum` to insert two paragraphs of text before your sentence, and two *different* paragraphs after your sentence.

### Solution 4.1

```
\documentclass[]{article}

\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{lipsum}

\begin{document}

\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle

\lipsum[1-2]

This is going to be an amazing document.

\lipsum[3-4]

\end{document}
```

## 4.6 Document Classes

The class of your document gives  $\text{\LaTeX}$  a set of defaults for formatting. Right now, we’re working with a document of the class ‘article’, which is probably the class you’ll use the most. This class is what you would use for any sort of short to medium length writing format: scientific journals, presentations, short reports, program documentation, invitations, etc. It is one of the base classes, and is quite flexible.

There are also several other base classes:

Class	Usage
article	For articles in scientific journals, presentations, short reports, program documentation, invitation
IEEEtran	For articles with the IEEE Transactions format
proc	A class for proceedings based on the article class.
report	For longer reports containing several chapters, small books, thesis
book	For real books
slides	For slides. The class uses big sans serif letters
memoir	For changing sensibly the output of the document. It is based on the book class, but you can create any kind of document with it
letter	For writing letters.
beamer	For writing presentations (see LaTeX/Presentations)

#### Challenge 4.2

Try changing the class of this document to some of the other classes to see how the formatting differs

#### Solution 4.2

Note that, among other things, many of the non-article classes have title pages by default

Throughout this tutorial, we’re going to continue using the article class, as it’s the most widely used in scientific publishing. However, we can instantly make this document have a more book-like beginning by moving the ‘top matter’ to it’s own page by wrapping it in a titlepage command:

```
\begin{document}

\begin{titlepage}
\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle
\end{titlepage}
```



### 4.6.1 Document Class Options

We are currently using all of the default options, but there lots of document class options available.

Option	Function
10pt, 11pt, 12pt	Sets the size of the main font in the document. If no option is specified, 10pt is assumed.
a4paper, letterpaper	Defines the paper size. The default size is letterpaper
fleqn	Typesets displayed formulas left-aligned instead of centered.
leqno	Places the numbering of formulas on the left hand side instead of the right.
titlepage, notitlepage	Specifies whether a new page should be started after the document title or not. The article class does not start a new page by default, while report and book do.
twocolumn	Instructs LaTeX to typeset the document in two columns instead of one.
twoside, oneside	Specifies whether double or single sided output should be generated. The classes article and report are single sided and the book class is double sided by default. Note that this option concerns the style of the document only. The option twoside does not tell the printer you use that it should actually make a two-sided printout.
landscape	Changes the layout of the document to print in landscape mode.
openright, openany	Makes chapters begin either only on right hand pages or on the next page available. This does not work with the article class, as it does not know about chapters. The report class by default starts chapters on the next page available and the book class starts them on right hand pages.
draft	makes LaTeX indicate hyphenation and justification problems with a small square in the right-hand margin of the problem line so they can be located quickly by a human. Suppresses the inclusion of images and shows only a frame where they would normally occur.

### Challenge 4.3

Make a title page for your article without using the `\begin{titlepage}...\end{titlepage}` by making a title page environment. Make the base font as large as possible, and make the body text into two columns.

### Solution 4.3

```
\documentclass[12pt, twocolumn, titlepage]{article}

\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{lipsum}
\usepackage{float}

\begin{document}

\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle

\lipsum[1-2]

This is going to be an amazing document.

\lipsum[3-4]

\end{document}
```

Notice that `\begin{titlepage}...\end{titlepage}` and `\documentclass[titlepage]{article}` have slightly different outputs

## 4.7 Organization

The keywords for organizing your document will differ depending on what class you are using, for instance only books and reports have chapters, however they all work the same way. L<sup>A</sup>T<sub>E</sub>X automatically numbers and formats each level correctly, so all you have to provide for each is a text title. The following table lists the levels from largest to smallest:

Command	Level	Comment
<code>\part{Part}</code>	-1	not in letters
<code>\chapter{Chapter}</code>	0	only books and reports
<code>\section{Section}</code>	1	not in letters
<code>\subsection{SubSection}</code>	2	not in letters
<code>\subsubsection{SubSubSection}</code>	3	not in letters
<code>\paragraph{Paragraph}</code>	4	not in letters
<code>\subparagraph{SubParagraph}</code>	5	not in letters

Although it's not listed here (because it's not part of the sectioning hierarchy), you can also add an abstract by enclosing the appropriate text in a `\begin{abstract}... \end{abstract}` statement.

### Challenge 4.4

Using the lipsum package to fill in text, give your paper an abstract and make two new sections in your document. In one section, add both a subsection of text and a sub-subsection.

#### Solution 4.4: Example answer

```
\documentclass[12pt, twocolumn, titlepage]{article}

\usepackage{lipsum}

\begin{document}

\title{Overleaf From Scratch}
\author{Amanda Charbonneau}
\date{\today}
\maketitle

\begin{abstract}
\lipsum[10]
\end{abstract}

\section{This is going to be an amazing document}
\lipsum[1-2]

\section{Some Background}
\lipsum[3-4]

\subsection{A diversion}
\lipsum[5-6]

\subsubsection{An Aside}
\lipsum[7]

\end{document}
```

Note the auto-numbering

Now that we have some sections to refer to, we can also add a Table of Contents using the `\tableofcontents` command. *L*<sup>A</sup>T<sub>E</sub>X *will* add the ToC anywhere you put the command, so be sure to put it somewhere sensible, like right after the `\maketitle` command (Or just after the abstract, if you have one). Generally speaking, you want to let *L*<sup>A</sup>T<sub>E</sub>X handle line and page breaks, however if you want the ToC to get it's own page, it's usually safe to

follow it with a `\pagebreak`

## A Note about Stars

For many commands, you'll see that there are two versions, for instance you can choose to make a `\section{}` or a `\section*{}`. In almost all cases, the non-asterisk version is the default behavior of the command, whereas the asterisk version is modified in some way. Often, the modification is a useful extension, for instance, `\section*{}` makes an unnumbered section that doesn't appear in the table of contents, however it's not always easy to guess the behavior of the non-default. Using the asterisk version generally won't hurt anything, so feel free to test it out. In this tutorial, unless I specify to use the `*` version, I'm just using the default.

## 5 Non-paragraphs

Now that we've got a basic document, lets add something besides paragraphs. Non-paragraph sections are special, and each is contained in it's own **environment**, so that each has a beginning and an end. All of these environments must still be inside your document environment, or they won't get rendered. Let's add each of these in ways you might use in a manuscript.

### 5.1 Lists

One of the easiest things to add is a list. Lists can be numbered, un-numbered, or definitions, and they all use the same command: `\item`. How the list is formatted depends on what type of list environment you wrap the list in:

```
\begin{enumerate}
\item taco
\end{enumerate}

\begin{itemize}
\item taco
\end{itemize}

\begin{description}
```

```
\item[taco] A delicious food
\end{description}
```

### Challenge 5.1

Make a new, un-numbered section between your abstract and current first section. Call the new section "Important Points". Inside the new section, make a short, bullet point list of why people should read your new manuscript.

### Solution 5.1: Example Solution

```
\begin{abstract}
\lipsum[10]
\end{abstract}
\section*{Important Points}
\begin{itemize}
\item Invasive species cost the U.S. over \$120 billion/year
\item At least 50,000 non-native species have been introduced
      to the U.S.
\item Invasive species are endemic to every continent except
      Antarctica
\end{itemize}

\section{The Origin of Invasive Species}
\lipsum[1-2]
```

## 5.2 Tables

There are lots of packages to make tables to exacting specifications, however basic tables are simple to make. Begin a tabular environment, and for each column, specify whether the text should be left (l), center (c) or right (r) justified as the options. If you want vertical separators between columns, those also go in the options as | or || for double lines. Inside the environment, separate column contents with a &, and end each line with a hard line return: \\

```
\begin{tabular}{l|c|r}
left & center & right\\
Words & Go & Here\\
\end{tabular}
```

You can add horizontal lines between rows (or anywhere else you like) by calling `\hline`

```
\begin{tabular}{l|c|r}
\hline
left & center & right\\
Words & Go & Here\\
\hline
\end{tabular}
```

### Challenge 5.2

Insert a table at the end of section 1. Make your table three left-justified columns wide, with vertical lines between each column and on the table edges. Also make a horizontal line separating the column headers from the data, and one at the bottom of the table. For this exercise, keep your names and entries short so they fit in the current column width. We'll learn how to make more elaborate tables shortly.

### Solution 5.2: Example Solution

```
\section{The Origin of Invasive Species}
\lipsum[1-2]

\begin{tabular}{|l|l|l|l|}

\hline
Population & Origin & Invaded & \\
\hline
MAES & Santander, Spain & Georgia, USA & \\
COAU & ?, Spain & Cowra, AU & \\
\hline
\end{tabular}

\section{Some Background}
\lipsum[3-4]
```

#### 5.2.1 Better Tables

Although this is clearly a table, it's rendered pretty poorly. That's because it's not a float, that is, it hasn't been given a specific space to exist in. To give it some space, and help L<sup>A</sup>T<sub>E</sub>X properly render it alongside text, we need to wrap it in a `table` environment. L<sup>A</sup>T<sub>E</sub>X has internal algorithms that find the optimal place to put float figures without ruining the flow of text, but you can alter this behavior by putting different float specifiers in a set of brackets `\begin{figure}[ ] ... \end{figure}`. Your options are: h, H, t, b, p, and !, which translate into "hereish", RIGHT HERE ON THIS SPOT, top of page, bottom of page, special page at the end, and "Just ignore all your logic settings and do what I say". They can also be combined.

Inside a float environment, you can also center with `\centering` and assign a caption with `\caption{}`.

#### Challenge 5.3

Wrap your table in a `table` environment, give it a caption, and get it to sit at the top of the second column of text.



### Solution 5.3: Example Solution

```
\lipsum[2]

\begin{table}[!t]
\begin{tabular}{|l|l|l|l|}
\hline
Population & Origin & Invaded & \\
\hline
MAES & Santander, Spain & Georgia, USA & \\
COAU & ?, Spain & Cowra, AU & \\
\hline
\end{tabular}
\caption{Invasive populations and their country of origin}
\end{table}

\section{Some Background}
```

## 5.3 Equations

L<sup>A</sup>T<sub>E</sub>X was written to handle mathematics, and can write pretty much any non-nonsensical equation that you can conjure up. As such, there's actually an overwhelming number of different math modes. We're only going to explore the most common ones, but if you have a specific math requirement, refer to the Mathematics, Advanced Mathematics, Theorems, Chemicals, Algorithms, Code and Linguistics pages of the L<sup>A</sup>T<sub>E</sub>X wikibook [5]. The amsmath package that we already loaded also allows L<sup>A</sup>T<sub>E</sub>X to recognize a surprising number of math words from commands in standard English. Because equations were the very expensive to typeset in the before times, to write bits of math in line, you wrap the statements in  $or  $\left( \right)$ .$

Let  $\phi = X_1, X_2, \ldots, X_n$

Note that by using the notation of  $X_2$  you can get mathematically formatted subscripts of any type. If you want the letters within your formula to be formatted as letters instead of numbers (so less italic) you can wrap them in `\text{}` inside of the math call.

Let  $\phi = X_a, X_b, \dots, X_c$ , where  $\phi$  is a sequence of independent and identically distributed random variables with  $\text{E}[X_i] = \mu$  and  $\text{Var}[X_i] = \sigma^2 < \infty$

To make equations on their own line, simply wrap the statement in double dollar signs (Apparently, equation lines are *extra* expensive), or `\[ \]`, for the  $\text{\LaTeX}$  equivalent. Many of the math commands also take variables, which is how you assign things like fractions and square roots.

$$S_n = \frac{X_1 + X_2 + \dots + X_n}{n} = \frac{1}{n} \sum_{i=1}^n X_i$$

### Nerd Aside: Which syntax is better?

Technically, the  $\$$  (and  $$$$  syntax) are  $\text{\TeX}$  syntax. The official  $\text{\LaTeX}$  syntax is `\( \)`.

If you look on the right part of the internet, you can find people who will tell you with great passion that using  $\text{\TeX}$  syntax in a  $\text{\LaTeX}$  environment is tantamount to murdering puppies. While it's technically true that one can conjure up edge cases where the  $\text{\TeX}$  version will give incorrect spacing and other crimes against kerning, it's also true that the people that get worked up about this sort of thing are also likely to be the ones you'd edge away from at parties. In practice, I've never seen a case where it mattered, no one I know has seen a case where it mattered, and no one they know has seen a case where it mattered. I'm pretty confident that it's easier to connect me to Kevin Bacon than to a reason to care. My advice? Use whatever you want. I like the  $\$$ , because it has an amusing connection to expenses that helps me remember it, but Overleaf has a handy button that will auto-insert the  $\text{\LaTeX}$  version of either in-line or own line equations. You do you.

#### Challenge 5.4

Insert these equations on their own line, between the paragraphs of section 1:

$$\begin{aligned} F &\sim U + S + V + \epsilon \\ &\sim X + UxP \end{aligned}$$

Hint: The tilde is called **sim**, and the error symbol is **epsilon**

### Solution 5.4: Example Solution

```
\section{The Origin of Invasive Species}
\lipsum[1]
```

```
\[F \sim U + S + V + \epsilon \]
\[\sim X + U \times P\]
```

```
\lipsum[2]
```

```
\begin{tabular}{|l|l|l|l|}
```

OR

```
\section{The Origin of Invasive Species}
\lipsum[1]
```

```
$$$ F \sim U + S + V + \epsilon $$$
$$\sim X + U \times P$$
```

```
\lipsum[2]
```

```
\begin{tabular}{|l|l|l|l|}
```

#### 5.3.1 Better Equations

While the inline and own line environments can be accessed with the dollar sign or bracket shortcuts, the rest of the math environments look more like the familiar `\begin{...}\end{...}` format. Two of these are `align` and `aligned`.

#### Challenge 5.5

Use the `align` environment to make put this equation at the beginning of section 1:

$$F \sim U + S + V + \epsilon$$

Question: How does using the `align` environment change the displayed equation?

### Solution 5.5: Example Solution

```
\section{The Origin of Invasive Species}

\begin{align}
F \sim G + S + D + V + \epsilon \\
\end{align}

\lipsum[1]
```

By using the `align` environment, the equation is automatically given a number. If we're writing a manuscript, we probably want all of the equations numbered.

### Challenge 5.6

Go back to the first, two line, equation that we used and wrap it in an `align` statement instead of `$$`.

### Solution 5.6: Example Solution

```
\begin{align}
F \sim U + S + V + \epsilon \\
\sim X + U \times P \\
\end{align}
```

Inside the `align` environment, two things happened, our two line equation was forced onto a single line *and* is given two equation numbers. We fixed the first by putting in a manual line break. However, we need to tell  $\text{\LaTeX}$  that our two lines are all one equation by wrapping them in *another* math environment: `aligned`

### Challenge 5.7

Try to get the two-line model equation to display as with a single equation number.

#### Solution 5.7: Example Solution

```
\begin{align}
\begin{aligned}
F \sim U + S + V + \epsilon \\
\sim X + U \times P
\end{aligned}
\end{aligned}
```

#### Challenge 5.8

At the end of section 2, use some of the column formatting you learned while making tables to re-create this algebra demonstration:

$$\begin{aligned} 2x + 3 &= 7 \\ 2x &= 4 \\ x &= 2 \end{aligned} \tag{1}$$

#### Solution 5.8: Example Solution

```
\begin{align}
\begin{aligned}
2x+3 &= 7 \\
2x &= 4 \\
x &= 2
\end{aligned}
\end{aligned}
```

## 5.4 Images

Let's say we want to add an image. This might be a fancy graph you made in R or SAS or a photo of a gel or experimental result. First, you need to get your image onto Overleaf. Overleaf supports .pdf, .jpeg, .jpg, .png, .eps, .epsf, .epsi, .pgf, and .tikz image files.

1. If you haven't already, load the `graphicx` package

2. Click the Project button to toggle open the Files column
3. Click on Add files
4. Click on Upload from... Computer
5. Drag in the images you want, or click Choose to open a file browser
6. If you will have many images, you can also use the Add files... New Folder drop down menu to put them in

Once the files are uploaded, Overleaf supports drag and drop rearrangement

7. In your document, `\includegraphics[width=\linewidth]{PATH-to-file}` at the desired graphic position.

You can choose widths other than `\linewidth`, but it's usually a good place to start

You can also set other parameters like height, scale, resolution and angle, among others. Just separate them with commas.

If you put your images in a new folder called "images" the PATH-to-file is `images/filename.jpg`

#### Challenge 5.9

Add an image of your own just before the text of section 2.

#### Solution 5.9: Example Solution

```
\section{Some Background}  
\includegraphics[width=\linewidth]{images/AFFR_32_12.jpg}  
\lipsum[3-4]
```

### 5.4.1 Better Images

Like with equations, you *can* drop in images without invoking a complete image environment, however it gives you very limited control over where the image ends up in the final text and can disrupt the text in weird ways. Like

with tables, generally, it's better to use the `figure` environment, so the figure is treated as a float. Also like tables, it takes the same extra bracket to assign a float specifier, and is allowed the same float options 5.2.1. Since this is a float, you can still center with `\centering` and assign a caption with `\caption{}`.

#### Challenge 5.10

Wrap the image we've already made in a `figure` environment, give it a bottom caption, and place it somewhere around section 2.1.

#### Solution 5.10: Example Solution

```
\section{Some Background}

\begin{figure}[ht]
\centering
\includegraphics[width=.8\textwidth]{images/AFFR_32_12.jpg}
\caption{AFFR plant 90 days post planting}
\end{figure}

\lipsum[3-4]
```

## 6 Referencing

We're going to cover four main types of references. The first two are ways to internally reference bits of your document, and the last two are citing outside resources.

### 6.1 Self Reference

One of the nicest things about  $\text{\LaTeX}$  is that it auto-numbers all of our sections, figures, equations and tables so that we don't have to keep track of them. How then, do we reference a table in the text, if we don't know what number it will end up with? There are two commands for this `\label{}` and `\ref{}`.

1. Use the `\label{}` command to mark the line you wish to reference back to. The text inside the `\label{}` is what you'll use to call it
2. When you wish to reference a marked line, use `\ref{}` to call the same text.

For instance, to refer to our model equation later in the text we would alter the equation lines to read:

```
\begin{align}
\begin{aligned}
F \sim U + S + V + \epsilon \quad \& \& \label{equ1} \\
\sim X + U \times P \\
\end{aligned}
\end{align}
```

and then somewhere else in our text, write a sentence like:

This sentence is about how important equation `\ref{equ1}` is.

#### Challenge 6.1

Set up a label/reference for the image we just inserted.

#### Solution 6.1: Example Solution

```
\section{Some Background}
\begin{figure}[h]
\centering
\includegraphics[width=.8\linewidth]{images/AFFR_32_12.jpg}
\caption{\label{AFFRimage}AFFR plant 90 days post planting}
\end{figure}
```

This sentence is about how important equation `\ref{equ1}` is.  
This sentence explains how I derived equation `\ref{equ1}`  
from `\ref{AFFRimage}`.

#### Challenge 6.2

Set up a label/reference for the table we made earlier.



## Solution 6.2: Example Solution

```
\lipsum[2]

\begin{table}[!t]
\begin{tabular}{|l|l|l|l|}
\hline
Population & Origin & Invaded & \\
\hline
MAES & Santander, Spain & Georgia, USA & \\
COAU & ?, Spain & Cowra, AU & \\
\hline
\end{tabular}
\caption{\label{InvTable}Invasive populations and their
country of origin}
\end{table}

\section{Some Background}
\begin{figure}[h]
\centering
\includegraphics[width=.8\linewidth]{images/AFFR_32_12.jpg}
\caption{\label{AFFRimage}AFFR plant 90 days post planting}
\end{figure}

This sentence is about how important equation \ref{equ1} is.
This sentence explains how I derived equation \ref{equ1}
from \ref{AFFRimage}; essentially, I calculated distances
from table \ref{InvTable}
\lipsum[3-4]
```

## 6.2 Footnotes

I don't use footnotes very much, but when I do I follow the word I want footnoted with the footnote command `\footnote{}`. Just write your note inside the curly braces,  $\text{\LaTeX}$  does lots of convenient things<sup>2</sup>

---

<sup>2</sup>Like auto-number them, and auto-place them on the correct page

### Challenge 6.3

Add a footnote to one of your sentences

### Solution 6.3: Example Solution

```
This sentence\footnote{I'm using the term pretty loosely
  here} is about how important equation \ref{equ1} is. This
  sentence explains how I derived equation \ref{equ1} from
  image \ref{AFFRimage}; essentially, I calculated
  distances from table \ref{InvTable}
\lipsum[3-4]
```

## 6.3 Bibliographies

A good scientific paper needs a bibliography. Just like standard bibliographies, each type of  $\text{\LaTeX}$  reference has a slightly different format, to account for the different meta-data each has, but they're pretty similar to each other. For instance, the article `\bibitem{}` looks like this:

```
@article{Xarticle,
  author   = "",
  title    = "",
  journal  = "",
  %volume  = "",
  %number  = "",
  %pages   = "",
  year     = "XXXX",
  %month   = "",
  %note    = "",
}
```

To use a given reference, you use the `\cite{}` command, where you put the citekey into the curly braces. It's important that each thing you're going to reference has a unique citekey, and generally they end up being something like a partial author name and year. You would cite the empty example above with `\cite{Xarticle}`. One of the best things about **Overleaf** is that once you have given it your list of references in either of the ways discussed below,

it will offer citekey suggestions everytime you use the `\cite{}` command, so you don't have to memorize all of them.

There are two ways you can build a bibliography in  $\text{\LaTeX}$  but I would only ever consider using one of them. The non-Amanda-approved way is to just add it to the end of your document as another environment, just before the `\end{document}` call:

```
\begin{thebibliography}{00}

\bibitem{b1}

\bibitem{b2}

etc...

\end{thebibliography}
```

where the 00, is how many digits  $\text{\LaTeX}$  needs to allot for references, so this allows up to 99.

### 6.3.1 A better bibliography

The downside of that, is that you'll have to type or copy in each bibtex listing, and if you want to use them in a different paper, you'll have to dig through this one to find the right lines to copy out. Furthermore, if one of your lines has an error, and you copy/paste that bit out to many different papers, you'll have to fix it in every copy. And that's annoying.

Much better, in my opinion, is to have a separate bibtex file, that contains only bibliography information. Then you can just upload this file to each of your Overleaf projects. Whenever you make changes, you can just push that new copy out to all your projects, without having to manually edit them all.  $\text{\LaTeX}$  will only put items in your works cited once you actually cite them, so feel free to make your bibtex have everything you *might* cite, even if it's unlikely.

To set up this kind:

1. Make a plain text file filled with all your bibtex listings
2. Upload that file to Overleaf
3. At the end of your main document, just before the `\end{document}`:

4. `\bibliographystyle{bmc-mathphys} % Style BST file`  
`\bibliography{bmc_article} % Bibliography file (usually`  
`'*.bib' )`

Changing the bibliography style will change both the style of your works cited, and the in-line citations

#### Challenge 6.4

Copy the following example references into a new bibliography file, and cite both of these somewhere in your paper.

```
@book{Ehara:1950t1,  
author = {Ehara, K and Abe, S},  
title = {{Classification of the forms of Japanese barnyard  
millet}},  
publisher = {Proceedings of the Crop Science Society of  
Japan},  
year = {1950}  
}  
  
@article{Yamane:2005bd,  
author = {Yamane, Kyoko and L{"u}, Na and Ohnishi, Ohmi},  
title = {{Chloroplast DNA variations of cultivated radish and  
its wild relatives}},  
journal = {Plant Science},  
year = {2005},  
volume = {168},  
number = {3},  
pages = {627--634},  
month = mar  
}
```

#### Solution 6.4: Example Solution

```
\section*{Important Points}
\begin{itemize}
\item Invasive species cost the U.S. over \$120 billion/year
\item At least 50,000 non-native species have been introduced
      to the U.S.\cite{Ehara:1950t1}
\item Invasive species are endemic to every continent except
      Antarctica\cite{Yamane:2005bd}
\end{itemize}
```

### 6.3.2 An even better bibliography

The base L<sup>A</sup>T<sub>E</sub>X citation doesn't have very many options. For instance, if you want your in-text citations to have the name and date rather than a number, you need something else. One of the most commonly used packages is `natbib`, which allows a variety of modifications to both your in-line citation and the works cited section. However, it can only use some of the built in style files, and generally you have to provide your own. On the bright side, if you're submitting to a journal, they'll probably provide you with a style file, it will be a `.bst` I've provided copies of the usage guide and three new `.bst` files with this project. They're all in the `natbib` folder.

In addition to the standard `\cite{}` command, `natbib` also has a `\citep{}` and `\citete{}` which let you switch between parenthetical and textual in-line citations. These commands also take several options, to change what is included in each inline reference.

### Challenge 6.5

Using `natnotes.pdf` as a guide do the following:

1. Load `natbib` so it will sort inline citations and separate multiples by commas
2. Change one of your current citations to a see parenthetical, i.e. (see Yamane et al., 2005)
3. Change one of your current citation to a textual reference, reword the sentence as necessary
4. Make at least one reference to multiple papers, by including multiple citekeys, separated by commas
5. Try one of the new `.bst` style files

### Solution 6.5: Example Solution

```
\item According to \citet{Yamane:2005bd, Ehara:1950tl}, at
  least 50,000 non-native species have been introduced to
  the U.S.
\item Invasive species are endemic to every continent except
  Antarctica\citep[see][]{Yamane:2005bd}

...Upload plainnat.bst to main folder of current project...

\bibliographystyle{plainnat}
\bibliography{bib.bib}
```

## 7 Collaboration

### 7.1 Sharing

Overleaf makes collaboration really easy.

- If you click on the **SHARE** button, you'll see a number of links you can use to send both editable and read-only versions to colleagues; or to


put into your git repository if you have one.

- The PDF button will automatically make and download a version of your document to PDF format.
- The PUBLISH button will let you automatically publish to an array of common document sharing websites like Figshare and ArXiv.


## 7.2 Version Control

### 7.2.1 Minor versions

Unlike a  $\text{\LaTeX}$  document that you make locally, you can share your Overleaf creations without worry. Overleaf has automatic version control, which means that everything you, or anyone else, types in your document is logged.

To access previous versions, click on the  button. This will bring up a new window with the history of your document. Each change will be labeled with who made the change, and the time since that change, as well as a clickable link to retrieve each previous version. Restoring a previous version will move it to the top of the versioning stack, but any versions between where you started and the one you chose are kept, so you can easily revert your reversion.

### 7.2.2 Major Versions

Another cool feature can be accessed by clicking the  button. This gives you a way to quickly revert to major versions. So, for instance, if you are about to send out your document for comments, you can name the current version as "Pre-Jeff". So even if, for instance, your collaborator flails and deletes everything, and has no concept of versioning, so he starts frantically re-writing it in a haphazard way before you notice...you don't then have to scroll back through hundreds of new versioning snippets looking for the original. In fact, you can have many named versions, so you can easily flip between old and new.

#### Challenge 7.1

Make our current document a major version

### Solution 7.1: Example Solution

click the  button, and type in a name

Another cool feature can be accessed by clicking the

## 7.3 Commenting

If you choose to share an editable link with others, they can offer comments even if they have no idea how to use  $\text{\LaTeX}$ .

Instructions for non- $\text{\LaTeX}$  users:

1. Go to web link provided
2. On the top left of your screen click **Rich Text** to make it green
3. Read the formatted text on the right side of the screen.

You can make the right side bigger or smaller by dragging the green arrows pointing left and right near the top of the screen
4. When you find a sentence where you'd like to leave a comment, click it with your mouse
5. The  $\text{\LaTeX}$  version on the left side of the screen will automatically scroll to the corresponding bit of code
6. Highlight the text you want to comment on the left side of the screen then click the comment button (it's the speech bubble next to **Rich Text**)
7. A box will pop up allowing you to leave your comment, just click **Comment** when you're done
8. If you want to directly change text, you can type as normal in the left side of the screen. It works a lot like Word. So if your cursor is in a numbered list and you hit enter, it will give you a new line with the next number.
9. All done? Everything is automatically saved as you do it

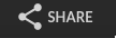


## 7.4 Legit Git

If you use Git, the versioning system of Overleaf probably looked pretty familiar. That's because each of your Overleaf documents is automatically initialized as a Git repo on Overleaf's private server.

While you don't have access to the repo directly, if you have Git installed locally, you can get a copy of it whenever you like.

### 7.4.1 Overleaf into a clean director

1. Click on the  button
2. Copy the Git Clone address
3. In a local terminal window, cd to where you want the repo to be saved
4. `git clone http://|address|`

This will initialize a new, local git repo, and clone in the entire version history of your Overleaf document.

Please note that you have cloned out the repo, so from this point forward, your Overleaf document and local document can evolve independently. They won't automatically sync changes. You can however sync any changes you make with the standard `git pull` and `git push` commands

## Part III

# Customization

## 8 Finer Text Control

*We're going to change a lot of the basic formatting in this document. If you want to keep the document you already have, just go to your Overleaf dash (by clicking the Overleaf logo on the top left of your screen), and on your current document, click the downward facing arrow, and then **Open a Copy**.*

Let's go back to our title page. Say we have a number of co-authors, we can add extra people to our author list by just manually adding line breaks:

```
\author{Amanda Charbonneau\\
Titus Brown}
```

However, that gives us equal weighting, and centering. What if instead I want to add a list of my committee members, but with their own heading, and without implying that they wrote my thesis? Unfortunately, the default title environment won't let us just add more author calls, or even re-arrange the order that the title elements come in. If we want to restructure the title page to meet specific requirements, we need to get out of easy mode and write it from scratch.

We want to use the `\begin{titlepage}...end{titlepage}` and get rid of all of the automatic commands (`\author{}`, `\date{}`, `\title{}`).

Most title pages are centered, so we want to make our title page environment center automatically:

```
\begin{titlepage}
  \begin{center}
    \end{center}
\end{titlepage}
```

Now everything we put between this pair of `\begin{}`...`\end{}` statements will be placed on the title page and centered. If you don't like centering, try `\begin{flushleft}` or `\begin{flushright}`, but strangely, not `\begin{align}` or any of its derivatives, as those are for managing equations, which was covered in 5.3

## 8.1 Fonts

### 8.1.1 Sizes

Now we just need to add the content, and tell  $\text{\LaTeX}$  how to render it. Even though we're writing our own title page, we aren't going to select absolute font sizes. That would defeat the purpose of having a flexible typesetting system. Instead, we're going to tell it the *relative* sizes, so our title page will still change based on our base font size. There are several font size keywords, from smallest to largest they are:

1. `\tiny`
2. `\scriptsize`

3. `\footnotesize`
4. `\small`
5. `\normalsize`
6. `\large`
7. `\Large`
8. `\LARGE`
9. `\huge`
10. `\Huge`

one two three four five six seven eight nine ten

These size commands continue until they are stopped, which is almost never what you want. So generally, you want to wrap the size command and the text it is effecting in curly braces as in `{\huge nine}`

### 8.1.2 Families

You can also alter the look of all or part of your document by changing the font family. Using the command `\fontfamily{fontname}\selectfont` immediately after `\begin{document}` will render the entire document in the selected family. Like sizes, fonts continue until stopped, and wrapping the call and intended text in curly braces will limit the font to only that text. This will often be ugly, but might be useful for long quotes or other special situations. Here are some available fonts:

### **Serif Fonts**

#### **Abbreviation**

#### **Font Name**

cmr	Computer Modern Roman (default)
lmr	Latin Modern Roman
pbk	Bookman
bch	Charter
pnc	New Century Schoolbook
ppl	Palatino
ptm	Times

### **Sans Serif Fonts**

#### **Abbreviation**

#### **Font Name**

cmss	Computer Modern Sans Serif (default)
lmss	Latin Modern Sans Serif
pag	Avant Garde
phv	Helvetica

### **Typewriter Fonts**

#### **Abbreviation**

#### **Font Name**

cmtt	Computer Modern Typewriter (default)
lmtt	Latin Modern
pcr	Courier

Challenge 8.1: Note: Don't worry about spacing yet

Using the relative font sizes provided, write a new title page that has a title, your name, today's date, your university affiliation, and the names of three collaborators (real or fictional), centered and in that order. The words can be whatever size, but the title should be relatively the largest, followed by your name, your university, your collaborators and the date.

## Solution 8.1

```
\begin{titlepage}
  \begin{center}
    {\Huge A New Beginning}\\
    {\LARGE Amanda Charbonneau}\\
    {\large \today}\\
    {\Large Michigan State University}\\
    {\Large Committee Members}\\
    {\normalsize Bob Marley\\ Whitney Houston\\ Rainbow Brite}

  \end{center}
\end{titlepage}
```

### 8.1.3 Spacing

We already know that  $\text{\LaTeX}$  ignores line returns in your source, and you may have found that it also doesn't accept more than one `\\` per line. In most circumstances, this isn't a problem, the horizontal spacing is automatic and looks fine—but this title page is a mess. There's two easy ways to change the spacing in parts of your document. The first is to load a package that allows you to vary the spacing in familiar ways. Adding `\usepackage{setspace}` to the preamble of your document will give you access to useful commands like:

1. `\doublespacing`
2. `\onehalfspacing`
3. `\singlespacing`

Or, you can set the spacing explicitly using `\vspace{#cm}`

## Challenge 8.2

Using the code you've already written, add vertical spacing between your title lines until you have something like:

A New Beginning

Amanda Charbonneau

Genetics Department  
Michigan State University

Submitted on  
January 4, 2016

Committee Members:  
Bob Marley  
Whitney Houston  
Rainbow Brite

## Solution 8.2: Example Solution

```
\begin{titlepage}
  \begin{center}
    \vspace{1cm}
    {\Huge A New Beginning}\\

    \vspace{2.0cm}

    {\Large Amanda Charbonneau}\\

    \vspace{.5cm}
    \begin{doublespacing}

    {\large Genetics Department}\\

    {\Large Michigan State University}\\

    \end{doublespacing}

    \vspace{2cm}

    {\large Submitted on\\
    \today}\\

    \vspace{2.0cm}
    {\Large Committee Members:}\\

    {\normalsize Bob Marley\\ Whitney Houston\\ Rainbow
    Brite}

  \end{center}
\end{titlepage}
```

## 8.2 Emphasis

Right now our text is okay, but wouldn't it be better if some of the words stuck out more than others? Like this?, **or this?**, *or this?* To get reliable text manipulation from L<sup>A</sup>T<sub>E</sub>X, regardless of what packages you may be using, you wrap the words you want to effect in the correct command: `\underline{}` to underline, `\textbf{}` to bold face, or `\textit{}` to italic face:

```
\underline{This is going to be an \textbf{amazing} document.}\\
\textit{It really is!}
```

You want to use these commands in places where the text needs to have *exactly* the manipulation you request, e.g. italicizing a species name. If, instead what you want is to emphasize a word, use `\emph{}`. In our current document, `\emph{}` and `\textit{}` are indistinguishable to the reader, however they're doing very different things behind the scenes. `\textit{}` always makes text italic, `\emph{}` makes the word stand out from whatever text is around it.

### Challenge 8.3

Copy out the following three lines and add emphasis to some of the words to see how it changes the text

```
\textit{I never said she stole my money}\\
\textbf{I never said she stole my money}\\
\underline{I never said she stole my money}\\
```

### Solution 8.3

*I never said she stole my money*  
**I never said she *stole* my money**  
I never said she stole *my* money

## 8.3 Colors

To add colors to your document, you need to load a package that defines colors. The most basic is `color` which gives you access to the named colors: white, black, red, green, blue, cyan, magenta, yellow. If you need more, there



are other color packages, like `xcolor` that will give you even more. If you need more than *that*, you can make up your own. As there are lots of places that you might want to produce color, there is a relatively large number of commands that colorize each document element individually.

```
\textcolor{declared-color}{text}
{\color{declared-color} some text}
\pagecolor{declared-color}
\colorbox{declared-color}{text}
\colorbox{declared-color1}{\color{declared-color2}text}
\fcolorbox{declared-color-frame}{declared-color-background}{text}
```

Note that for some color packages, you need to specify that you want to call colors by name in the options as in:

```
\usepackage[usenames,dvipsnames,svgnames,table]{xcolor}
```

#### Challenge 8.4

Make a small box with some text in your document, as if you were writing a review paper.

#### Solution 8.4: Example Solution

`fcolorboxdeclared-color-frameddeclared-color-background`  
**Box 1. Important Weed Things** There are many important things to know about weeds

## 8.4 Defining your own commands

### 8.4.1 Custom colors

There are several ways to define your own colors, all using the same command `\definecolor{name}{model}{color-spec}` in the preamble of your document.

`name` is whatever you want to name your color

`model` is the way you describe the color, and is either `gray`, `rgb`, `RGB`, `HTML`, or `cmyk`.

color-spec is the numerical value of the color

Examples:

```
\definecolor{midgray}{gray}{0.65}  
\definecolor{orange}{rgb}{1,0.5,0}  
\definecolor{orange}{RGB}{255,127,0}  
\definecolor{orange}{HTML}{FF7F00}  
\definecolor{orange}{cmyk}{0,0.5,1,0}
```

### 8.4.2 Arbitrary Custom Formatting

In addition to defining new colors, you can also easily define a custom format using the `\newcommand{name}{variable}{formatting}` command.

name is how you want to call your command

variable is any variable used in the formatting options

formatting is calling a set of commands that will comprise your custom command

So if, for instance, you were going to frequently want *blue italic font*, you could define a custom command in your preamble like this:

```
\newcommand{\BIF}[1]{\textit{\textcolor{blue}{1}}}
```

and then throughout your paper, rather than continually write

```
\textit{\textcolor{blue}{words I want slanty and blue}}
```

you'd simply call

```
\BIF{How awesome am I?}
```

This has two immediate advantages: First, it's shorter and easier to type, and second, because it is defined in your preamble, you can instantaneously change every instance of custom fonts in your document by changing only a single line of code.

## 9 Custom Images

Frankly, I do all my graphing in R and other figures in OmniGraffle, output high quality graphs, import those graphs into Overleaf, and then display them as floats. However, it *is* possible to have L<sup>A</sup>T<sub>E</sub>X draw all sorts of plots and images directly. To make images like shapes and flow charts, you use a package called `tikz`. For graphing, you combine `tikz` with another called `pgfplots`

I'm including a couple of examples, however, mastering these packages is a tutorial all on it's own. If you're interesting in directly coding images, I highly recommend:

TexAmple.net A site with a gallery of example code[2]

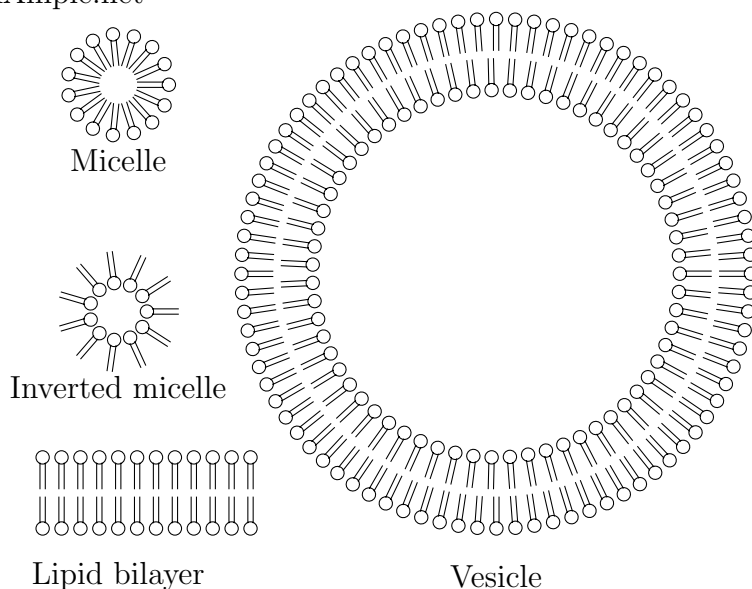
sharelatex:TikZ A short tutorial to making basic image parts[4]

A very minimal introduction to TikZ A slightly longer tutorial[1]

sharelatex:pgf A short tutorial to basic graphs[4]

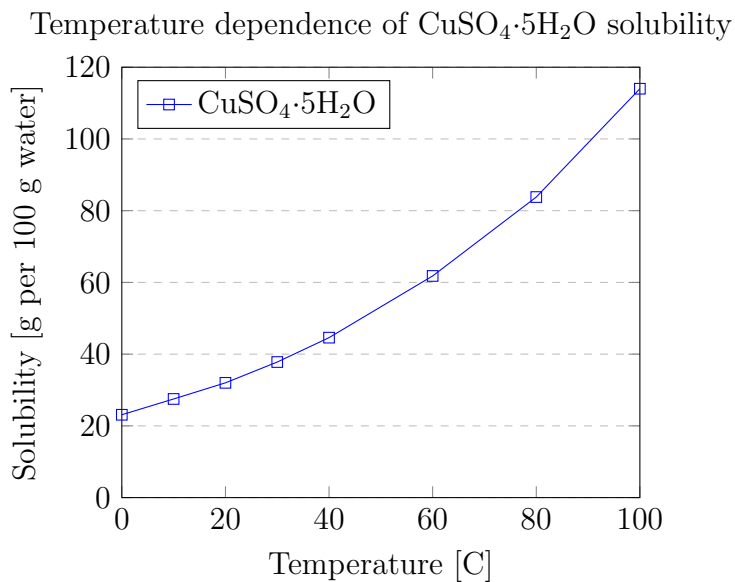
### 9.1 Images

An example of lipid membranes, drawn with `tikz` shamelessly stolen from TexAmple.net



## 9.2 Graphs

An example plot, drawn with `tikz` and `pfgplots`, shamelessly stolen from sharlatex:pgf[4]



## References

- [1] J Cr mer. *A very minimal introduction to TikZ*. Manuscript 0 (0), 2011.
- [2] Kjell Magne Fauske. [TexAmple.net](http://www.texample.net).
- [3] D E Knuth. *The Future of TEX and METAFONT*, 1990.
- [4] H Oswald, J Allen, and B Gough. *ShareLaTeX, the Online LaTeX Editor*.
- [5] Wikibooks. *LaTeX. Markup Language*. opensource-books, April 2014.