# GETTING STARTED WITH LaTeX AND VIM

JACKSON VAN DYKE

## Contents



I wrote this document to archive the process of setting up my workflow for taking notes and writing papers in case I needed to reference it for some reason in the future. I figured there's no reason not to share it, in case someone might find it useful. This is by no means exhaustive, and is meant as a jumping-off-point. The actual documentation of the various software/packages should be the reader's next stop. The `.tex` file for this document is available at this repository.

If you just want my style files, these can found at this repository, and if you just want my vimrc, this can be found at this repository.

This guide is written primarily for a mac, though most of what is written here would work fine for any Unix machine.

## 1. Introduction

LaTeX is a very simple typesetting tool which creates, in my opinion, very visually appealing documents. Much of the generic appeal of LaTeX is that a `.tex` document comes in the form of a plaintext document which is somehow modular. One sense in which this is true is that it can be compiled using any 'document class'. This means that if you decide to write a paper hoping to submit it to one journal which wants the paper to look one way, and later decide to submit the paper to another journal which wants it to look another way, the structure of the actual pdf document can be changed immediately. This is just one example of the many things which make

*Date*: March 18, 2020.

LaTeX so great. For more information and a much better technical introduction to how this thing actually works, I suggest the canonical reference, *The Texbook*, available here.

Vim is a text editor which uses keystrokes (rather than menus and the cursor) to navigate and edit text documents. In other words you use combinations of keys on your keyboard to toggle through lines, copy, paste, find, etc. It can be used from a command line interface or as a standalone applications in a graphical user interface. So you can edit documents using Vim directly in the terminal or by opening the Vim application as you would open Microsoft Word. The fact that it uses keystrokes makes it hard to learn, but much faster once you do. It is also highly customizable with what it called a vimrc. We will learn about this in section 3.2.

## 2. Setting up LaTeX

2.1. **Installation.** Download from, and follow instructions from this webpage.

2.2. **Macros.** LaTeX macros should be thought of as custom 'settings'. A basic example of such a thing is the following:

```
\newcommand{\RR}{\mathbb{R}}
```

which says that whenever one types `\RR` it is interpreted as `\mathbb{R}`. All such rules are collected in what is called a style file (a file with ending `.sty`) which lives in the directory

```
~/Library/texmf/tex/latex/
```

or the directory of whatever file you are editing. This is also where class files live. See this repository for mine.

## 3. Setting up Vim

3.1. **Installation.** Vim should come with your machine, but to make sure you have the latest version, or to install it for the first time, it is easiest to use homebrew. I.e. enter the following to the command line:

```
brew install vim
```

3.2. **Writing a `vimrc`.** The content of a `vimrc` file should be thought of as custom settings for Vim. When starting out it is tempting to copy and past someone else's. If you would like to do this, mine can be found at this repository. I would however suggest building up your own gradually, adding things as you need them. Google is helpful for this. By default your vimrc just lives in `/User/` but I prefer to have it in `~/.vim` so it is easier to push to by backup repository.

3.3. **Packages for Vim.** There are countless packages available for Vim. I mention the main ones I use. Consult the documentation directly for more information.

(1) Pathogen: In short, this package creates a directory such that whenever a package is placed in it, the package is loaded. There are many alternatives to this.
(2) Nerd commenter: This package lets one comment lines quickly. It detects the document type so one doesn't have to keep track of which symbols comment lines in which document type, and instead one can have a designated keystroke combination which comments lines in any language.
(3) Spell: Offers spell checking, and spell correction.

(4) Ulti-Snips: Snippets are far too complicated to get into in depth, but they basically allow you to type an abbreviation for something, hit a special key (by default `<tab>`) and this abbreviation will expand to the larger predesignated thing. As one can imagine, this is extremely useful for typing things quickly. Especially in LaTeX. To write and customize snippets:

```
:UltiSnipsEdit texmath
```

(5) Vimtex: This is, in my opinion, the best package supporting writing LaTeXwith Vim. The alternative, Vim-LaTeX, is much heavier and isn't as flexible. In combination with Ulti-Snips, vimtex can do much more.

(6) Vim-tex-fold: Supports folding of a LaTeXdocument in Vim. This means that, for example, sections will be collapsed to a single line on ones display when they are not being edited.

## 4. Displaying the PDF

When editing notes quickly, it is useful to let the compiler run every time the document is saved. This is the default mode in vim-tex. This means the PDF viewer one is using must change as the file changes. Most PDF viewers do not support this. In my opinion there are only two effective options for PDF viewers on a mac that support live updating: Skim and Zathura.

4.1. **Using Skim.** To install Skim enter the following into the command line:

```
$ brew cask install skim
```

4.2. **Using Zathura.** To tap, install, and link Zathura/the required plugins enter the following into the command line:

```
$ brew tap zegervdv/zathura
$ brew install zathura --with-synctex
$ brew install zathura-pdf-poppler
$ brew install xdotool
$ mkdir -p $(brew --prefix zathura)/lib/zathura
$ ln -s $(brew --prefix zathura-pdf-poppler)/libpdf-poppler.dylib
        $(brew --prefix zathura)/lib/zathura/libpdf-poppler.dylib
```

4.3. **Making a choice.** If using vim-tex, once you have determined which viewer you would like to use, add one of the following lines to your `vimrc`:

```
let g:vimtex_view_method='zathura'
let g:vimtex_view_method='skim'
```

and the pdf file will automatically open in your chosen viewer when it is compiled. Use `\ll` to compile a document in vim-tex. See the documentation of vim-tex for more information on PDF viewer support and compiling documents.

## 5. Writing a document

This section assumes my `vimrc` and LaTeXmacros are being used. The preamble and document environment (specifically for taking notes) is triggered by

```
notes<tab>
```

with Ulti-Snips placeholders (toggled through with `<c-j>`) at all of the places where things should be added.

To suggest a completion of a word:

`<ctrl>n`

To correct spelling:

`<ctrl>p`

To add a word to the dictionary:

`zg`

To remove a word from the dictionary:

`zug`

To reset folds:

`\zx`

To fold current fold:

`\za`

Toggle whether current line is commented:

`\c_`

Build an environment out of whatever is in the current line:

`<ctrl>b`