# Introduction to PlainText

Installation Guide — macOs

Timothy Elder

PlainText
Working
Group

# Installation

## LaTeX, pandoc, and `Git`

### LaTeX

LaTeX, typically pronounced *lay*-tek or *lah*-tex, is a program for typesetting documents developed in the late 70s. It allows you to create really pretty documents and in particular, documents that include mathematical notation, figures, and tables. It can be used to typeset pretty much anything that needs typesetting.

LaTeX is a pretty big program (around 5GBs) so you will need sufficient space on your hard drive to accommodate it. Depending on your system you will need to navigate to the install page and pick the version that is appropriate for your OS. Follow the installation instructions as you would any other program. This takes a little time due to its size.

### `pandoc`

`pandoc` bills itself as a "universal document converter" and can be used to convert files into different formats. It was primarily developed to take documents written in `markdown` and convert them into other, prettier formats. When combined with LaTeX, `pandoc` is a pretty powerful tool and we can use it to write in `markdown` and then simultaneously convert that document into Word, PDF, Tex and HTML. It is this software that allows you to integrate your work with your colleagues that might not want to make the shift to a plain text workflow. Further, you can actually take in documents in a common format like Word and convert them to plain text.

Just like for LaTeX above, navigate to the install page, and download the installer appropriate to your OS. `pandoc` is significantly smaller than LaTeX so you should not have an issue installing it.

### `Git` and Github

Git is a version control system which monitors a directory for changes. Think of it as the "Track Changes" option for a Word Document, but it tracks all the changes that occur to a group of files in a directory. Git is a program with a command line user interface so you will need to know your way around a terminal to use it. It also allows you to take advantage of open source software freely available on Github.

GitHub is a website that hosts repositories of software, and if you sign up for a free account, acts like a cloud backup for your projects when they are in plaintext. All the software hosted on GitHub can be installed using `Git`, and we will use it to install the templates we are going to be using for our documents. Install `Git` here.

## Installing Document Templates

Now that you have LaTeX and `pandoc` installed you are ready to start using plain text software for your research and writing. But if you want to have nicely formatted documents that handle all the bells and whistles that you need for your scientific and social scientific writing then you need to do just a few more steps. This is the harder part of the installation process but I will take you step by step through it.

### What are we installing?

We are going to be installing two collections of files: the first is a set of templates that we use to take in our raw plain text files and typeset them into beautiful PDF, HTML, and (to satisfy your Word dependent colleagues) Word Documents. These templates are what `pandoc` uses to make pretty output, but the templates themselves rely upon what are called LaTeX class and style files. These are the second collection of files we are installing. A quick preview of how to use this software. When you are done writing your document or you want to see what it ultimately will look like, you will go to your terminal and tell `pandoc` essentially "take my plain text markdown file and make it into a PDF document" by typing in a few commands.

`pandoc` then converts the plain text data (the prose, tables, and figures you've written and created) into a PDF using LaTeX as the software to typeset it. It isn't the case that LaTeX understands what markdown is, but `pandoc` is super nifty and actually does an intermediate step that you don't see where it converts your markdown formatted plain text file into a LaTeX file before then typesetting it. Because these are, strictly speaking, two different processes which are done by two different pieces of software that are talking to one another, we need to install the templates into two different places on your computer, where LaTeX and `pandoc`, respectively, look for files.

We are going to be installing the two collections of templates for typesetting documents from my GitHub page using `Git`. This is actually a very straight forward and easy process but feels really overwhelming at first. All you need to do is open a terminal, `cd` to the place that we want the template files to be and tell `Git` to download them.

The first set of templates is in a repository called pandoc-templates and includes the templates that take in our markdown files and convert them into LaTeX files. The second is a set of document class and style files that include all the details for how the software will typeset our documents with LaTeX. These can be found in the repository latex-custom-kjh. Finally, we will also install a template directory, md-starter, which includes template markdown and `Make` files which you can use for pretty much any document. Don't worry about what these are yet, we will cover that shortly when we start using them.

Ensure that that the software you need is installed by running the following code in a terminal:

```
pandoc --version

latex --version

git --version
```

Each of these commands, if the software is installed, will print out version information. If you the software is not installed then your terminal will say something like `python: command not found` or `latex: command not found` which means something has gone wrong and you will have to attempt to install the software again. ***NOTE***: Go to the appendix below for learning out how to add to your PATH variable which could solve your problem for you.

You need to make sure that you create a directory and a sub-directory the custom LaTeX templates, class and style files. Run the following code (changing the paths for your machine):

```
mkdir /Users/timothyelder/Library/texmf && mkdir /Users/timothyelder/
    Library/texmf/tex/ && mkdir /Users/timothyelder/Library/texmf/tex/latex
```

This creates a directory where you install the custom class and style files for LaTeX. With those directories made you can install the templates into their proper locations with the following code (again, make sure you change the paths for your machine):

```
cd /Users/timothyelder
git clone https://github.com/timothyelder/pandoc-templates/
mv pandoc-templates .pandoc

cd /Users/timothyelder/Library/texmf/tex/latex
git clone https://github.com/TimothyElder/latex-custom-kjh/
```

If you did everything above right, Congratulations! You are one step closer to making beautiful and reproducible documents in plain text. The template and style files we installed will handle typesetting our documents so they appear the way we want, but there are a few other features of academic papers that we might want to include that requires two more pieces of software.

For one, you will likely be mounting an argument or writing in an area of your discipline that requires you reference past works. So you will need a bibliography, and you will probably want to include tables, figures and diagrams that you will reference in the body of your text. It is a pain having to format a bibliography yourself or to number each table and figure. What more, it is particularly annoying when you decide to rearrange your paper and then have to rearrange the number of all the tables and figures in the text. You wll also want a nice interface to actually write and code in.

## pandoc-xnos

`pandoc-xnos` is for cross-referencing figures, tables, equations, sections and pretty much anything else that can be written in a Markdown file. This helps for automatically handling the labelling of different parts of your document, so you can move the position of a figure or table or equations without having to change the reference to it in the body of your text. Install with:

```
pip install pandoc-fignos pandoc-eqnos pandoc-tablenos \
            pandoc-secnos --user
```

And then use it with `--filter pandoc-xnos` as a flag for the `pandoc` command in your `Makefile` recipe. **NOTE**: Whenever using `pandoc-xnos`, they need to be invoked before the `--citeproc` filter as they both use the @ symbol to identify references and they will get confused otherwise.

## Visual Studio Code

One of the advantages of plain text is that you can open your files on any computer with the native programs installed on it from the factory. Every computer will have a basic text editor (macOs has TextEdit and Windows has Notepad) and you could do everything we are going to do in this working group with the command line and one of these text editors (in fact you could do everything just with the command line). I *highly* recommend you install a more advanced text editor to do your work in.

There are a few excellent options to choose from including Sublime Text, Emacs, Atom, Notepad++ and RStudio. By far the best, and the one I use is Microsoft's Visual Studio Code (or VS Code for short). The advantage to using a text editor regardless of which specific one you choose is that they highlight the syntax of whatever programming language you are writing in (including Markdown and LaTeX) enhancing the readability of your code to help writing and debugging. What more, you can typically run everything directly in the text editor so you never need to navigate away from a single window to get all your work done. You can run `R`, `python`, and keep your LaTeXdocument open all at the same time toggling between the different tasks as you need.

You should install VS Code as it is what I will be using to demonstrate how to use these plain text tools. You can install it from here. One of the nice things about VS Code is that it is *extensible* and has all sorts of extra features designed and implemented by users to help you get things done. One of the most important extensions is a LaTeX helper that we will be using when we do have to edit or write in LaTeX. After you install VS Code go ahead and install the extension here. There are also other helpful things for writing like a spell checker, a tool for auto-completing citations, a word counter, and support for your favorite (or not so favorite) statistical software. And don't worry, installing these extensions is just a matter of clicking a button.

## Conclusion

Congratulations! You have actually just done a lot of the work of getting started using plain text for your research and writing. This is an onerous task and you should be commended for your effort. If you have managed to get this far then you are likely somewhat invested in adopting the plain text paradigm but if this effort has inspired skepticism or reticence about diving deeper I think it is important to note that installing the software is something that you only have to do once on any given computer. So you won't have to endlessly tinker with things. Once these tools are installed they are very dependable.

# Adding to PATH

A lot of the work of typesetting is going to be taking place in the terminal and sometimes you will get an annoying an inexplicable error like “this software is not on the PATH” or “I can’t find this software”, which means that your computer is searching the location where executable programs are located (the PATH variable) but it is not finding whatever software you are telling it to run. Why this happens is a mystery. Sometimes the software has installed onto your computer, but the terminal program doesn’t know where to look for where the programs are installed. I know this is a little confusing but here is a quick lesson. When you invoke a command on the terminal the first thing the computer does is check an index of the available software, which it does by examining what is called the PATH variable. If it finds whatever you are asking it to use like `pandoc` or LaTeX or R or python or whatever, it runs the command. If it doesn’t find what you tell it to look for it returns an error.

Before giving up completely try updating the PATH variable so the terminal knows where the software is. Depending on what specific command line or terminal you are using the instructions are slightly different but they will be pretty portable across platforms, and operating systems. This is one of those things that you really only have to learn once and then you’ll be able to improvise a lot better later. To find out where the files for a program are on macOs, run the `which` command followed by what your looking for, such as `which pandoc`, and it will print the path to where the executables of these files are installed. Sometimes the data files for a program get installed onto our computer without the PATH variable getting updated to tell the terminal that the software is installed.

There are two different kinds of terminals on macOs. Run this line of code in your terminal and it will tell you which terminal you are using: `echo $0`. Check below for relevant information about how to add to the path:

## `zsh`

To add a directory to your path in zsh: 1. Run `path+=('/path/to/dir')` in an open terminal where the path is to where the executables for the program are. For me `pandoc` is located in `/usr/local/bin/pandoc` and LaTeX is in `/Library/TeX/texbin/latex`. 2. Then run `export PATH` in the same terminal. 3. Restart the terminal and check that it works by typing in `latex --version` or `pandoc --version` or whatever else was giving you trouble.

## `bash`

Remember when we learned how to use `vim` from the command line and what a hidden file was? Well now that knowledge will actually come in handy.[1] To add something to the PATH in `bash` you need to edit your `.bash_profile` which is located in your home directory. The home directory is where your terminal opens up and is located at `/Users/your-user-name`.

1. Open the `.bash_profile` file in your home directory (for example, `/Users/your-user-name/.bash_profile`) in `vim` by running `vim .bas_profile` in your home directory in a terminal.
2. Add `export PATH="your-dir:$PATH"` to the last line of the file, where *your-dir* is the directory you want to add.
3. Save the `.bash_profile` file by exiting `vim` with `:wq`.
4. Restart your terminal.

[1] If you are unfamiliar with `vim` and hidden files, refer to the session 1 documentation, "Project Organization and the Terminal".