Introduction to PlainText

Installation Guide

Timothy Elder

PlainText
Working
Group

February 17th, 2023

Installation

LaTeX, pandoc, and Git

LaTeX

LaTeX(typically pronounce *lay*-tek or *lah*-tex) is 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.

If you have limited disk space you can try installing a lightweight version of LaTeX called TinyTex but I have not used this version so attempt its use with caution.

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 softeware freely available on Github.

GitHub is a website that hosts repositories of software, and if you signup 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

We are going to be installing two collections of templates for typesetting documents from GitHub using `Git`. 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.

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

```
pandoc --version

latex --version

git --version
```

To find out where these are installed on either Windows or macOs, run the `where` command followed by what your looking for, such as `where pandoc`, and it will print the path to where the executables of these files are installed.

You need to make sure that you create two directories for installing these custom LaTeX templates and the LaTeX class and style files. Run the following code (regardless of your OS):

```
mkdir /Users/timothyelder/.pandoc # For pandoc

mkdir /Users/timothyelder/Library/texmf && mkdir /Users/timothyelder/
    Library/texmf/tex/ mkdir /Users/timothyelder/Library/texmf/tex/latex #
    For LateX
```

The first line of code creates a directory where you will put all your pandoc templates and the second is where you install the custom class and style files for LaTeX. Don't worry about what these are yet, we will cover that shortly in the Usage section.

With those directories made you can clone the repos into their proper locations with the following code (there are two versions depending on your OS):

For macOs

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

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

For Windows

```
cd C:\Users\Timot\AppData\Roaming

git clone https://github.com/TimothyElder/pandoc-templates.git # pandoc
    templates
ren "pandoc-templates" ".pandoc"

cd C:\texmf\tex\latex
git clone https://github.com/TimothyElder/latex-custom-kjh.git # latex
    style files

cd C:\Users\Timot\Documents
git clone https://github.com/timothyelder/md-starter # project template
```

Make sure that you change all the paths to match your machine otherwise you get lots of errors and nothing will work.

Bibliography and Cross-Referencing Management

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 bibligraphy, and you will probably want to include tables, figures and diagrams that you will refernce 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.

So you have installed pandoc and LaTeX, you installed Git and you cloned the repositiories that we are going to be using. naturally and so make sure they are installed, but also on two further repos: pandoc-crossref and pandoc-citeproc-preamble which I have much less of a command over.

install pandoc-crossref with Homebrew with:

```
brew install pandoc-crossref
```

There is a warning about the need for the crossref library/package to be built on the same verison of pandoc that you have installed but we will see what happens.

For the `pandoc-citeproc-preamble` install you will need Haskell and its cabal package manager thing.

Install cabal package manager with Homebrew (ghcup and Haskell are way too big):

```
brew install cabal-install
```

Then install `pandoc-citeproc-preamble` with:

```
cabal install pandoc-citeproc-preamble
```

pandoc-fignos

`pandoc-fignos` is for numbering and labeling figures in a LaTeX like way, so you can move the position of a figure without having to change the reference to it in the body of your text. Install with:

```
pip install pandoc-fignos --user
```

And then use it with `--filter pandoc-fignos` as a flag for the `pandoc` command in your `Makefile` recipe. **NOTE**: Whenever using `pandoc-fignos` or `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.

Adding to PATH

A lot of the work of typesetting is going to be taking place on the command line and sometimes you will get an annoying error like “this software is not on PATH”, 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.

Now depending on what specific comamand 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.

This is an error that you are almost certainly going to get with the software that handles cross referencing and bibliographic management

Choosing a Text Editor

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 workshop 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. I use Microsoft's Visual Code which is particularly good and seems to be becoming the standard. 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 workdone. You can run `R`, `python`, and keep your LaTeXdocument open all at the same time toggling between the different tasks as you need.

Supplement for Windows installation

First a moment for evangelism. Windows is far and away the most popular operating system in the world for desktops and laptops, with 76% of the global market share, followed by macOs (16%) and then Linux (5%). Pretty much everyone has a working familiarity with Windows, and what more it is meant to be user friendly for the average person. But as I articulated above, Windows offers a much more What You See Is What You Get Approach to computing. Window's User-Friendliness comes at the cost of controlling every part of your operating system. What more, Apple products are more cost prohibitive than products that use Windows, and Linux can be intimidating as it has comparatively less support than the more popular alternatives. With those caveats, macOs and Linux (both UNIX based operating systems) are comparatively better development platforms for the kind of computing tasks we will be covering in this workshop. For one, their file structure is much more intuitive and includes fewer redundancies, so finding where to install the repos we are working with is going to be achieved much quicker on macOs or Linux than on Windows.

But with all that said see below for instructions for installing this software on Windows.

Path to Tex location for custom files `C:\Users\Timot\texmf\tex\latex`

Path to pandoc location for custom templates: `C:\Users\Timot\AppData\Roaming\pandoc`

For Installing pandoc templates on Windows

```
cd C:\Users\Timot\AppData\Roaming

git clone https://github.com/TimothyElder/pandoc-templates.git

ren "pandoc-templates" ".pandoc"

start .pandoc
```

Some of the paths in the files in this repo will need to be changed, in particular the paths in your `Makefile` that identify where your bibliography is located and where the pandoc templates are installed. For me I had to change `/Users/timothyelder/.pandoc/` to `C:\Users\Timot\AppData\Roaming\pandoc`

For Installing Latex templates on Windows

To find where to install run `kpsewhich -var-value=TEXMFHOME`

```
cd C:\texmf\tex\latex

git clone https://github.com/TimothyElder/latex-custom-kjh.git
```

Now if we have installed the the custom latex files into the proper place (wherever your Tex distribution looks for custom files which is, I admit, a complete mystery to me when it comes to windows) then you wont need to update any of the paths in this directory. `/Users/timothyelder/Library/texmf/tex/latex/latex-custom-kjh -> C:\localtextmf\tex\latex\latex-custom-kjh`

From Tex installation manual

Specifying Additional Input Directories The command-line option `--include-directory=dir` causes the program to include dir into the list of input directories.

For example:

`latex --include-directory="C:\My Styles"foo.tex` This prepends `C:\My Styles` to the input search path, i.e., `C:\My Styles` will be searched first, when TeX tries to find an input file.

Also see this page for info about where to put the style files and then updating where MikeTex looks for custom style files.

Linux installation instructions:

If you are working on Linux then you could probably teach this workshop…