

Writing Technical Papers with Markdown

Sunday, Dec 20th 2015

Academic writing involves :

- ▶ writing down ideas as they come along and documenting results (notetaking),
- ▶ experimenting with these ideas (simulations and data analysis)
- ▶ and finally presenting them effectively (scientific paper)

A word about Word

- ▶ Word does not work in the workflow for **scientific research papers or reports**.



Figure 1: Raymond Hettinger

\LaTeX - lah-tekh, lah-tek or lay-tek

Enter \LaTeX .

\LaTeX is to a book what a set of blueprints is to a building.
[5]

Essentially, \LaTeX is a markup language. Content is written in plain text and can be annotated with commands that describe how certain elements should be displayed.

For example, take a look at the following commands.

```
\textbf{bold}
```

```
\textit{italic}
```

This markup will format the words passed into these “functions” as **bold** and *italic* respectively.

`\section{Section Name}`

This is text in the section

`\subsection{Sub Section Name}`

The following is a list in this subsection

`\begin{enumerate}`

`\item The first \textbf{bold} item`

`\begin{enumerate}`

`\item Nested item 1`

`\item Nested item 2`

`\end{enumerate}`

`\item The second \textit{italicized} item`

`\item The third etc \ldots`

`\end{enumerate}`

Markdown

Markdown is a very lightweight easy-to-read easy-to-write plain text markup language. The same example as before looks like this in Markdown.

```
# Section Name
```

```
This is text in the section
```

```
## Sub Section Name
```

```
The following is a list in this subsection
```

- ```
* The first bold item
 - Nested item 1
 - Nested item 2
* The second italicized item
* The third etc ...
```

| Right | Left  | Center | Default |
|-------|-------|--------|---------|
| ----- | ----- | -----  | -----   |
| 12    | 12    | 12     | 12      |
| 123   | 123   | 123    | 123     |
| 1     | 1     | 1      | 1       |

Table: Demonstration of simple table syntax.

This is what the same table looks like in L<sup>A</sup>T<sub>E</sub>X.

```
\begin{longtable}[c]{@{}rlcl@{}}
\caption{Demonstration of simple table syntax.}
\tabularnewline
\toprule
Right & Left & Center & Default\tabularnewline
\midrule
\endfirsthead
\toprule
Right & Left & Center & Default\tabularnewline
\midrule
\endhead
12 & 12 & 12 & 12\tabularnewline
123 & 123 & 123 & 123\tabularnewline
1 & 1 & 1 & 1\tabularnewline
\bottomrule
\end{longtable}
```

However, Markdown does not allow for the level of detailed customization that you can achieve using  $\text{\LaTeX}$ . Even a moderately complex table such as the one below is not supported (currently) by any form of Markdown.

|             |             |
|-------------|-------------|
| 7C0         | hexadecimal |
| 3700        | octal       |
| 11111000000 | binary      |
| 1984        | decimal     |

Figure 2: Tabular  $\text{\LaTeX}$  example [6]

Markdown may not be as powerful as  $\text{\LaTeX}$ , but its easy to write easy to read syntax, open standard format and a strong backing from the community make it a ideal candidate for writing. It has the advantages of Word (ease of use) and  $\text{\LaTeX}$  (excellent typesetting) for output formats. Also there is the added advantage of only having to write in Markdown once, and have documents generated in a multitude of formats later - PDF, DOCX, slides, HTML etc.

## Pandoc - A “swiss army knife”

Pandoc is a software tool written in Haskell that can convert a document from just about any format to just about any other format. And works really well.

## Pandoc conversion

To generate a PDF file :

```
pandoc document.md -o document.pdf
```

With PDF files, you can specify the following additional arguments :

- ▶ `--latex-engine=pdflatex` : latex engine
- ▶ `--latex-template=latex.template` : latex template file



With html files, you can specify the following arguments:

- ▶ `--template=html.template` : html template file
- ▶ `--css=cssfile.css` : css file

With docx files unfortunately, you cannot specify a template (at least not at the time of writing this post) [16]. You can however, specify a reference-docx :

- ▶ `--reference-docx=reference.docx` : docx for reference styles

These following arguments allow you to use citations when writing academic papers.

- ▶ `--filter pandoc-citeproc` : filter to parse citations
- ▶ `--csl=CSLFILE` : define a citation style sheet e.g. `ieee.csl`
- ▶ `--bibliography=BIBFILE` : look for citations from a bibliography

Also, I've found the following filters useful.

- ▶ `--filter pandoc-eqnos` : equation numbers
- ▶ `--filter pandoc-fignos` : figure numbers
- ▶ `--filter pandoc-tablenos` : table numbers

They allow you reference a figure, equation or table. For example, Equation 1 is an example of a block equation in Markdown.

## Downside to using Markdown?

The good news is that anything you do in  $\text{\LaTeX}$ , you can do in Markdown and render as a PDF. This includes equations, tables, citations, references, images, lists, tikz diagrams etc. The bad news is that if you do decide to use  $\text{\LaTeX}$  syntax, you are still writing  $\text{\LaTeX}$  (although a lot less of it), and you have lost complete HTML and DOCX conversion capability.

## Bending Markdown to your will

Fortunately, some of the problems I mentioned in the previous section can be solved using an excellent feature of Pandoc - filters!

There is even a python package called `pandocfilters` that allows you to walk the AST and parse specific formats or keys. It is very powerful, and can offer unique ways to expand on pandoc's functionality. I wrote a pandocfilter [19] to embed a jupyter notebook using a liquid tag style syntax, which I currently use for this post.

# TLDR

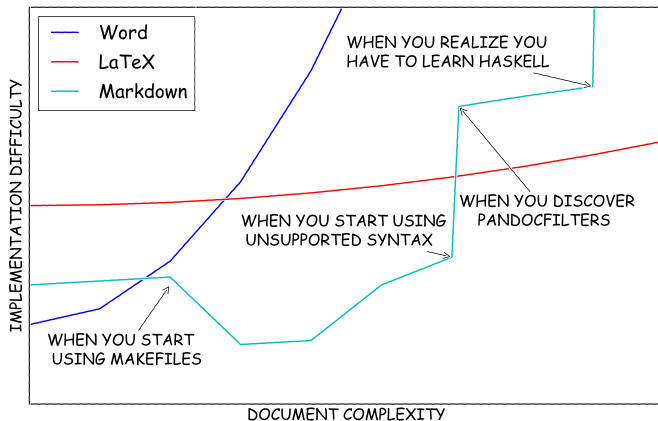


Figure 3: My very scientific comparison of Word,  $\text{\LaTeX}$  and Markdown