Pandoc for TeXnicians

John MacFarlane

TUG 2020, 2020-07-26

Contents

Overview

- What is pandoc?
- Using pandoc to convert to and from LaTeX
- Why write in Markdown?
- Overcoming Markdown's limitations

What is pandoc?

https://pandoc.org

Let's take it for a spin

```
% cat simple.tex
\section{On $e=mc^2$}\label{einstein}
```

```
% pandoc -f latex -t native simple.tex
% pandoc -f latex -t html simple.tex
% pandoc -t html --mathml simple.tex
% pandoc -t html --mathjax simple.tex
% pandoc -t -html --mathjax -s simple.tex
% pandoc -t ms simple.tex
% pandoc -t gfm simple.tex
% pandoc -t context simple.tex
```

```
% pandoc -t jats simple.tex
```

Some math

Let's try with a sample TeX document by Professor A.J. Roberts at the University of Adelaide (CC licensed).

http://www.maths.adelaide.edu.au/anthony.roberts/LaTeX/Src/maths.tex

Some math

```
% pandoc maths.tex -o maths.docx
```

. . .

Two problems:

- the use of a low-level TeX primitive `\mathcode`.
- the use of `\parbox` (line 288)

Fix by removing the `\mathcode` stuff and redefining the `\parmath` macro as a no-op:

```
\newcommand{\parmath}[2][]{#2}
```

Take two

```
% pandoc maths.tex --number-sections -o maths.docx
% open maths.docx
```

- AMS theorem environments come out right, including references.
- Math is translated into native Word equation objects, which can be edited and which match the font, rather than images.
- Still missing: equation numbers.

Going the other way

```
% pandoc maths.docx -o newmaths.tex -s
% xelatex newmaths
% xelatex newmaths
```

Converting to HTML

```
% pandoc maths.tex -s -o maths.html --mathml \
  --number-sections --toc
% open maths.html
```

Comparison with latex2rtf

```
% latex2rtf maths.tex
% open -a "Microsoft Word" maths.rtf
```

- References not resolved in Section 1
- Accents in Section 2 not above the letters, math generally ugly
- Arrays in Section 8 totally broken; same with subequations in Section 9
- But at least we do get equation numbers in Section 9

Comparison with tex4ht

```
% make4ht maths
% open maths.html
```

- Theorem environments not handled in Section 1 (except for one?).
- Missing accents in Section 2.
- Ugly equations that incorporate both text and images in different fonts.

Comparison with Word from PDF

```
% pdflatex maths
% pdflatex maths
% open -a "Microsoft Word" maths.pdf
```

- Section 2, accents messed up.
- Some formulas are rendered with images, others with regular characters, in non-matching font.
- The 'where' in Section 6 is badly mispleacd.
- The integral is missing in Section 7
- The diagonal ellipses are missing in the arrays

Pandoc can interpret TeX macros

```
% cat macros.tex
\newcommand{\nec}{\Box}
\newcommand{\if}[2]{#1 \rightarrow #2}
\newenvironment{warning}%
 {\begin{quote}\textbf{WARNING!}}%
 {\end{quote}}

$\if{\nec \phi}{\phi}$
\begin{warning}
Don't try this at home.
\end{warning}
```

```
% pandoc macros.tex -t html
```

Pandoc can resolve bibtex citations

With the help of the `pandoc-citeproc` filter (included in the released binaries).

```
% pandoc --filter pandoc-citeproc bib.tex \
  -t plain --csl ieee.csl
```

Limitations

Pandoc is far from being able to convert arbitrary tex files with high accuracy.

Let's try with a real-world example I got at random from arxiv.

```
% cd arxiv.2007.07694v1
% pandoc arxiv.tex -o arxiv.docx
```

An alternative

An alternative

So you can't just write in LaTeX and expect to convert at the last minute to docx (for a publisher) or epub (for your students) or HTML (for your website).

An alternative: write your document in pandoc's extended version of Markdown, which pandoc can convert with complete accuracy to any of its output formats.

What is Markdown?

Markdown is a set of conventions for indicating document formatting in plain text, mostly inherited from the pre-internet days of bulletin boards and email.

It was designed in 2004 by John Gruber with help from Aaron Schwartz, and it is currently much used by programmers, and on forums like stackoverflow and reddit, and by data scientists via Jupyter notebooks and RMarkdown.

https://daringfireball.net/projects/markdown/

Appealing things about Markdown

The source text is readable as it is. When writing and revising, you don't have to parse through command-words which aren't part of the content.

. . .

If you're writing in a language other than English, you don't have to have English words sprinkled in the text.

. . .

There's no boilerplate at the beginning. The document just starts with the text.

Real separation of content from formatting.

> The paucity of means is the greatest virtue of markdown and pandoc markdown.
>
> It is strangely difficult to get people to see the point, but the defects of LaTeX for concentration, writing and thought, are at least as great as those of Word, for the simple reason that it gives the writer too much power; there is always another package to call in the preamble, as there is always another drop down menu in Word. ...
>
> In markdown - not to put too fine a point on it - the writer is only ever faced with one question, and it is the right one: what the next sentence should be.
>
> — Michael Thompson, pandoc-discuss mailing list

Appealing things about Markdown

Using Markdown makes it possible to collaborate with others who don't know LaTeX.

Appealing things about Markdown

Markdown can be converted with complete, reliable accuracy into many different formats.

It's often not enough just to produce a PDF.

- JATS for publication or archiving
- EPUB for convenient reading on mobile devices
- Docx or ICML for a publisher
- HTML for a website (or accessibility)
- Jupyter notebook for research
- Beamer or reveal.js slides for presentation

TeX is a great assembly language for publication-quality documents.

Limitations of Markdown

John Gruber's original markdown syntax lacks support for:

- ☐ tables
- ☐ figures
- ☐ footnotes
- ☐ definition lists
- ☐ ordered lists other than decimal-numbered
- ☐ super/subscript
- ☐ math

- ☐ document metadata
- ☐ attributes or metadata on individual elements like sections
- ☐ labels and cross-references
- ☐ numbering for running examples or equations

Limitations of Markdown

We couldn't live without these things in academic writing.

And we definitely couldn't live without

- ☐ bibtex/biblatex
- ☐ macros

How can we overcome these limitations?

Overcoming Markdown's limitations

Pandoc's extended Markdown syntax

- ☒ tables (limited)
- ☒ figures (limited)
- ☒ math
- ☒ footnotes
- ☒ definition lists
- ☒ more flexible ordered lists
- ☒ running example lists
- ☒ super/subscript
- ☒ strikeout
- ☒ metadata
- ☒ attributes
- ☒ generic containers

Pandoc also understands LaTeX macro definitions, which you can use for math (no matter what the output format).

Labels and cross-references are still a work in progress, but you can get good support for them using an external filter, `pandoc-crossref`, by pandoc contributor Nikolay Yakimov.

You can use the `--citeproc` filter to resolve citations in this syntax:

```
Blah blah [@putnam:empirical, p. 33; see also
@dummett:empirical].
```

Change the style by specifying a CSL stylesheet. (You can even change between author-date, numerical, and footnote sytles with no modifications to the source.)

You can use your existing bibtex or biblatex bibliography file, or a CSL JSON bibliography such as can be produced by Zotero.

LaTeX macros allow you to define new constructions that exactly fit what you're writing about. Can we recover this flexibility?

Raw TeX in Markdown

One approach is to just include bits of raw TeX in your markdown file. Pandoc allows that.

- There is a special syntax for indicating chunks of raw TeX, but pandoc will also recognize obvious bits of raw TeX and pass them through as such.
- The raw TeX chunks will be passed on unchanged if the output format is `latex`, `beamer`, or `context`, and otherwise simply omitted.

```
% cat raw.md
% pandoc raw.md -o raw.pdf
% open raw.pdf
```

But:

```
% pandoc raw.md -s -o raw.html
% open raw.html
```

Drawbacks:

- With this approach you lose the ability to target multiple formats.
- Your source is now an ugly mix of Markdown and TeX, compromising readability.

A better approach

1. Adopt the convention that a certain thing representable in pandoc's markdown should be interpreted as, say, a dropped capital letter.
2. Write a filter that does the interpretation.

Example: drop caps

In LaTeX we can use the `lettrine` package to get dropped capitals at the beginning of chapters:

```
\lettrine{T}{his} is a pulley
```

We will use a generic bracketed span with a class to represent this in Markdown:

```
[This]{.dropcap} is a pulley.
```

Example: drop caps

Now we need a filter that replaces `Span` elements with class `dropcap` in the Pandoc AST with something appropriate for the output format.

Two kinds of filters

- **JSON filters** operate on a serialized JSON representation of the pandoc AST. They can be written in any language that can consume and produce JSON.
- **Lua filters** use a Lua interpreter and environment built into pandoc. No external software need be installed, and the filters are more efficient, because we don't need to serialize and deserialize as JSON.

Documentation: https://pandoc.org/lua-filters.html

Example: drop caps

In a Lua filter we define functions that match different kinds of AST elements. Here we want to match a Span. Create a file `dropcap.lua`:

```
function Span(el)
  -- do something with the Span (el)
  -- return the transformed element or a new element
end
```

Example: drop caps

We only want to do something if the Span has the class `dropcap` and its contents begin with a Str element.

```
function Span(el)
  if el.classes:includes('dropcap') then
    return make_dropcap(el.content)
  end
end
```

Example: drop caps

Now we just have to define `make_dropcap`. It takes a list of Inline elements (`el.content`) and returns a list of Inline elements.

```
local function make_dropcap(els)
  if els[1] and els[1].t == 'Str' then  -- arrays start at 1!
    local first_letter, rest = els[1].text:match('(%a)(.*)')
    if FORMAT == 'latex' then
      els[1] = pandoc.RawInline('latex',
                 '\\lettrine{' ..  first_letter ..
                 '}{' .. rest .. '}')
    elseif FORMAT:match('html') then
      els[1] = pandoc.Span({
                   pandoc.Span(pandoc.Str(first_letter),
                    {class='dropcap-first'}),
                   pandoc.Span(pandoc.Str(rest),
                {class='dropcap-rest'})})
    end
    return els
  end
end
```

Example: drop caps

```
% pandoc -L dropcap.lua -t latex -o dropcap.pdf
% pandoc -L dropcap.lua -t html -s --css dropcap.css \
    dropcap.md -o dropcap.html
```

Example: tikz diagrams

To get a tikz diagram, we could have a filter turn specially marked code blocks into images.

In fact, there is already a very nice general diagram filter at https://github.com/pandoc/lua-filters.

```
% cat diagram.md
% pandoc diagram.md -L diagram-generator.lua -s \
  --extract-media=media -o diagram.html
% pandoc diagram.md -L diagram-generator.lua \
  -o diagram.docx
```

Example: theorems

How to reproduce LaTeX `theorem` environments?

Markdown version:

```
::: {.theorem #pythagoras}
#### Pythagoras's Theorem
In a right triangle, the lengths of the two shorter sides
```

```
$a$, $b$ and the longer side $c$ stand in the relation
$$
a^2 + b^2 = c^2.
$$
:::
```

Example: theorems

```
% cat theorem.lua
% cat theorem.md
% pandoc -L theorem.lua theorem.md -t latex
% pandoc theorem.md -L theorem.lua -t plain
% pandoc theorem.md -L theorem.lua -t rst
% pandoc theorem.md -L theorem.lua -t html
```

The end

- For pandoc questions, come to pandoc-discuss on google groups: https://groups.google.com/g/pandoc-discuss
- For bug reports, the tracker at https://github.com/jgm/pandoc
- If you'd like to improve pandoc's handling of LaTeX, we can always use new contributors!

Questions?