plaintext_workshop

A workshop on Scientific Writing using plaintext tools like LaTeX, Markdown & pandoc

February 21, 2017, 12:00pm—5:00pm, NSC 245A

Why plaintext?

- separates content from formatting
- universally readable
- platform independent
- future-proof
- free and open source
- can easily use version control (e.g. Git & GitHub)
- can easily compare old vs new (e.g. opendiff (FileMerge) on OS X, e.g. Meld on Linux)

There are many blog posts out there about the virtues of using plaintext to do scientific (or more generally, academic) writing. Here are some of them:

- Sustainable Authorship in Plain Text using Pandoc and Markdown
- Academic Writing With Markdown
- Why (and How) I Wrote My Academic Book in Plain Text
- Markdown vs Latex for Academic Writing
- Writing Technical Papers with Markdown

Software

If you're on MacOS it would be a good idea to install Homebrew, a package manager that gives you easy access to installing all sorts of useful GNU/Linux tools including some of the tools listed here.

First let's **install LaTeX**. It's a large download. If you're on a Mac, install LaTeX using the downloadable binary installer. If you're on Ubuntu or another debian-based GNU/Linux, `sudo apt-get install texlive-full`.

- LaTeX
- MacOS: MacTeX
- GNU/Linux: TeX Live
- Windows: MiKTeX

Next let's **install pandoc** and some extras.

If you're on a Mac, and if you've installed homebrew, you can install pandoc using: `brew install pandoc pandoc-citeproc pandoc-crossref`. On GNU/Linux, I'm sure you can figure it out. I think pandoc is available using `sudo apt-get install pandoc` but I'm not sure about pandoc-citeproc and pandoc-crossref. Some googling should bring you an answer.

- pandoc

You will need a good **text editor** as well. There are many to choose from. Personally I use Emacs and Sublime Text. For our purposes here we don't need much sophistication so it comes down to personal preference.

Here is a list of some text editors:

- Emacs
- Vim
- Sublime Text
- Atom
- BBEdit
- Notepad++
- Visual Studio Code
- Gedit

There are some GUI-based LaTeX editors as well, if you're into that sort of thing (I'm not):

- TeXmaker
- TeXstudio
- TeXworks

There are also online LaTeX editors/compilers/environments. The advantage is that you don't have to install LaTeX locally on your own machine, and you don't have to worry about installing packages and updating outdated packages (they are all in the cloud). The disdvantage is that you can't do anything if you're not connected to the internet.

- ShareLaTeX
- Overleaf

There are also many, many Markdown-specific editors. This means they are meant for editing Markdown in particular, and they come with various kinds of built-in smarts for pretty-formatting Markdown.

Here are some:

- iA Writer
- Marked
- MacDown
- Bear
- Ulysses
- Typora
- Mou

If you're using Sublime Text (as I am) there is a nice Markdown plugin called MarkdownEditing.

Markdown

Let's start with Markdown, rather than LaTeX, since Markdown is less intimidating.

Markdown is a specification for "marking up" plain text documents using not-terribly-difficult-or-offensive codes that denote semantic elements. By "semantic elements" I mean things like headings, sub-headings, quotes, and so on. There are also variants of Markdown, such as Git-Markdown, that add various other codes to further extend Markdown's capabilities.

This document that you're reading (`README.md`) is a Markdown document. If you're reading it on GitHub, it's been rendered for you in your web browser by GitHub.

Here are some Markdown references online:

- Markdown: The original specification by John Gruber
- Mastering Markdown: a brief tutorial by GithHub

Here is a very simple Markdown document:

```
# Chapter 1

It was a bright cold day in April, and the clocks were striking
thirteen. Winston Smith, his chin nuzzled into his breast in an effort
to escape the vile wind, slipped quickly through the glass doors of
Victory Mansions, though not quickly enough to prevent a swirl of
gritty dust from entering along with him.
```

We have a heading, denoted using the # symbol, called "Chapter 1", followed by a paragraph of plain text. There are lots of other elements we can denote using Markdown, including things like lists, images, quotes, code listings, and so on. See the documentation for examples.

Note that there is no *stylistic* specification within a Markdown document. The codes denote semantic elements not stylistic elements. This is part of the philosophy of separating content from style. The idea is that one can take a semantically coded plain text Markdown document, and convert it into another format, such as HTML, or pdf, or a host of other formats, and along the way, specify a particular *style*—that is, a sort of translation of what each semantic element should look like.

If you've coded in HTML and used CSS files, this is the same idea.

The program we are going to use to perform this conversion is **Pandoc**.

Pandoc

Pandoc is a program that can convert not just from Markdown to a host of other formats, but can convert from multiple formats to multiple formats. The list of formats pandoc knows about is long.

Here are some resources for working with pandoc:

- Getting started with pandoc
- Pandoc Demos
- Pandoc FAQs
- Pandoc Manual
- Pandoc crossref for numbering figures, equations, tables and cross-references to them
- Pandoc citeproc for citations & bibliographies
- Citation Style Language (CSL) citation styles

Let's take the sample Markdown document above (the opening paragraph of George Orwell's "Nineteen Eighty Four") and convert it, using pandoc, to another format.

First save the document in a plain text file, let's call it `1984.md`. Now let's convert it into HTML format using pandoc:

```
pandoc 1984.md -o 1984.html
```

The file we get looks like this:

```
<h1 id="chapter-1">Chapter 1</h1>
<p>It was a bright cold day in April, and the clocks were striking
thirteen. Winston Smith, his chin nuzzled into his breast in an effort
to escape the vile wind, slipped quickly through the glass doors of
Victory Mansions, though not quickly enough to prevent a swirl of
gritty dust from entering along with him.</p>
```

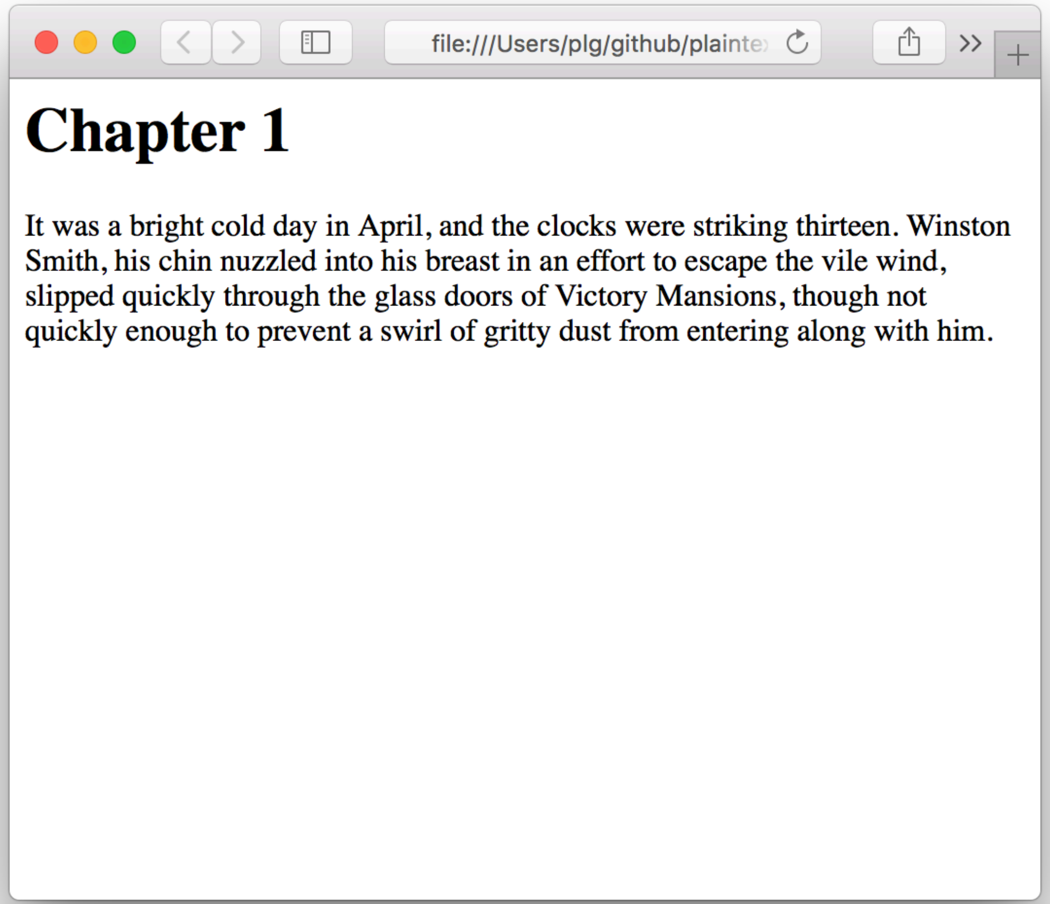
What you can see is that pandoc has basically translated the Markdown semantic codes into the corresponding HTML codes. In fact if you double-click on the `1984.html` file, and open it in a web browser, it will render the html file for you in the browser:

This sample document isn't very interesting, at least it doesn't make use of very many Markdown features. Let's have a look at another example that's closer to what we might be doing as academic writers:

```
# Methods

Participants grasped the handle of an IMT2 two degree of freedom robot
(InMotion Technologies Inc.) as they reached from a start position to
a movement target, located 20 cm away. The robot applied a
velocity-dependent force to the hand during movement, according to
[@eq:forcefield].

$$
    F_{x} = k \left[ v_{y} \right]
$$ {#eq:forcefield}

In [@eq:forcefield], $x$ and $y$ are lateral and sagittal directions,
$F_{x}$ is the applied robot force in the left-right direction,
$v_{y}$ is hand velocity in the forward-backward direction and $k$=14
Ns/m.
```

Here we have a heading `# Methods` followed by some text, and then an equation, denoted using `$$` codes to start and then end the equation. The equation is specified using LaTeX syntax. After the final `$$` code that denotes the end of the equation, we have a Markdown code that *labels* the equation using a label of our choice, in this case `forcefield`. What this enables us to do is refer to this equation using the *label*, and let Pandoc figure out the equation numbering. This is great, it means if we have many equations, we don't have to manually number them.

Save the above Markdown text in a plain text file called `robot.md` and let's convert it using pandoc to pdf format.



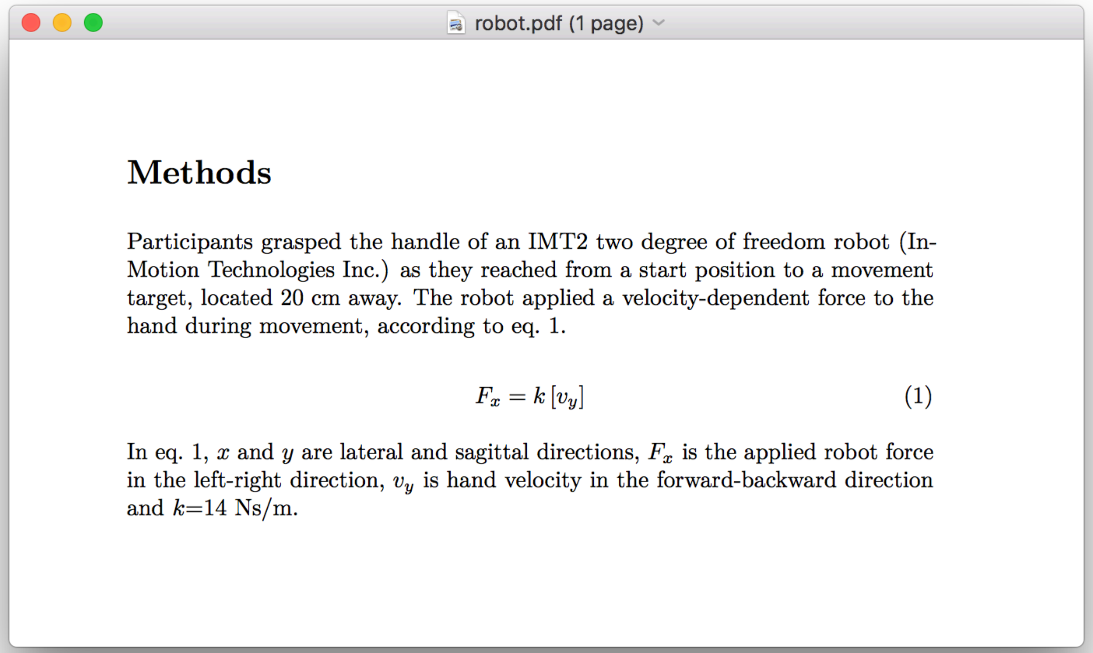
Chapter 1

It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

Figure 1: 1984.html

```
pandoc robot.md --filter pandoc-crossref -o robot.pdf
```

Now open the file `robot.pdf` and you will see something like this:



Methods

Participants grasped the handle of an IMT2 two degree of freedom robot (InMotion Technologies Inc.) as they reached from a start position to a movement target, located 20 cm away. The robot applied a velocity-dependent force to the hand during movement, according to eq. 1.

$$F_x = k\,[v_y] \tag{1}$$

In eq. 1, x and y are lateral and sagittal directions, F_x is the applied robot force in the left-right direction, v_y is hand velocity in the forward-backward direction and k=14 Ns/m.

Figure 2: robot.pdf

Pandoc actually uses LaTeX under the hood to convert from Markdown to pdf. We get a Heading ("Methods") in large, boldface font. We get a paragraph typset using a serif font (LaTeX uses a font called Computer Modern by default), and full-justified. We get our equation centered and numbered.

Note in the pandoc command above we used the flag `--filter pandoc-crossref`. This is needed in order to handle the equation labels and numbering properly.

The `[@eq:forcefield]` code we used to refer to our equation has been properly converted into "eq. 1". This is really useful once you have many equations. When you decide to reorder them, or delete some, or add some, you don't have to worry about maintaining the correct numbering. LaTeX does this for you. It also does this for Figure numbers, Table numbers, bibliographic references, table of contents, even an Index if you want one in your document.

Changing the font

Let's say you're not too fond of Computer Modern as a font choice. We can change that.

When Pandoc converts to pdf, it uses LaTeX as an in-between. In fact, Pandoc uses a LaTeX *template* to generate a LaTeX file, which is then converted, using LaTeX, to pdf. You can acutally see what the template looks like by typing:

```
pandoc -D latex > template.latex
```

and then opening the `template.latex` file you just created. It looks crazy if you've never seen LaTeX code before, and probably still looks icky even if you have. Pandoc has inserted a number of *variables* into the LaTeX document that lets you control various aspects of the final pdf output, using command-line arguments to Pandoc.

So for example to change the font to Helvetica, and to use 12pt instead of the default Computer Modern 10pt font, we can issue the Pandoc command like so:

```
pandoc robot.md \
--filter pandoc-crossref \
-V mainfont=Helvetica \
-V fontsize=12pt \
--latex-engine=xelatex \
-o robot.pdf
```

Now our document looks like this:

Our font is now Helvetica and the size is 12pt.

In the Pandoc command above, we use the `-V` flag to tell Pandoc to set a *variable* to a particular value. For example the `mainfont` variable is set to `Helvetica`, and the `fontsize` variable is set to `12pt`.

There is a whole bunch of variables that can be sent to Pandoc in order to control the look of your output, here is the relevant section of the Pandoc documentation:

Pandoc variables for LaTeX

Note that when changing the font away from one of the LaTeX standard choices (which are few, and arguably ugly), we have to tell Pandoc to use the `xelatex` engine instead of the standard LaTeX engine, to generate the pdf file. This is done with the command-line flag `--latex-engine=xelatex`. The standard LaTeX engine for generating a pdf is called `pdflatex` but it doesn't handle non-default fonts well. The `xelatex` engine does, so any time we want to change font away from the LaTeX defaults, I would suggest using this `--latex-engine=xelatex` flag. The `xelatex` engine allows you to use any font that's installed on your system.

xxx

other document changes (margin, etc, check out the variables list)

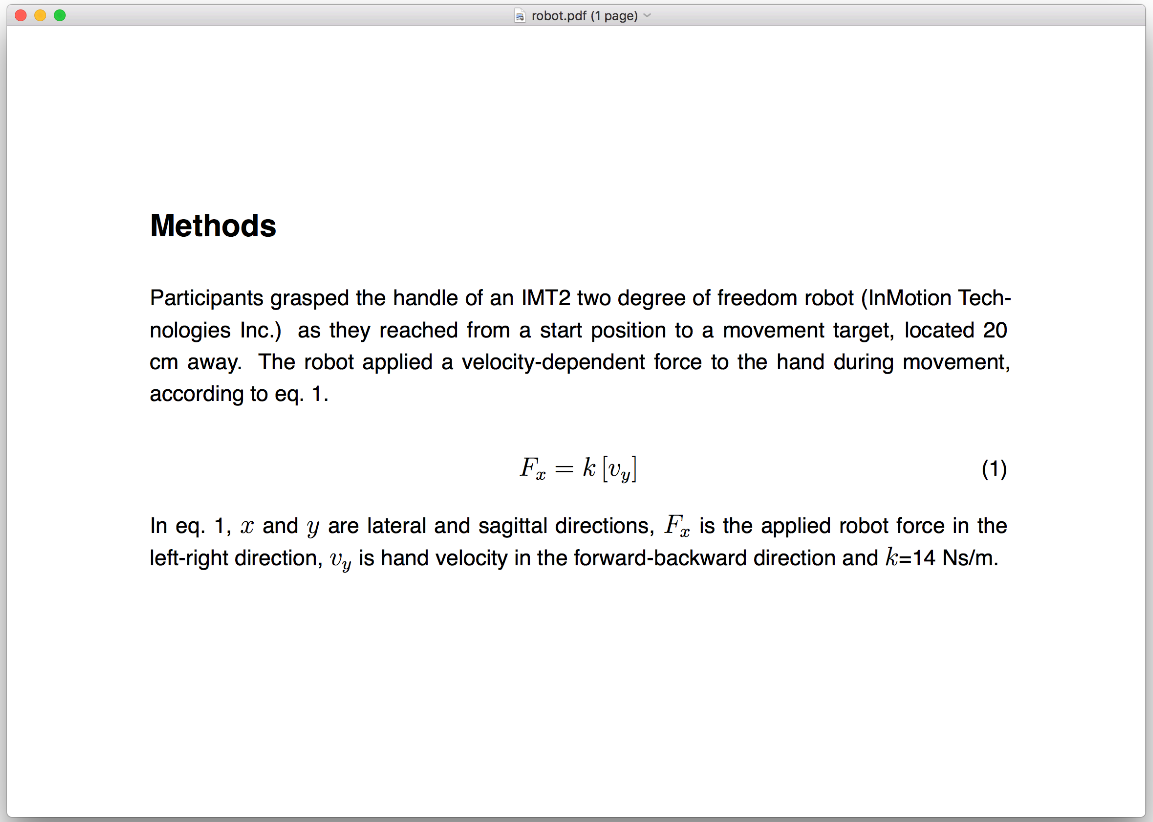
xxx

Makefile

xxx

Here is the command I use to convert this `README.md` document into a pdf file:

```
pandoc README.md \
-V geometry:margin=1.0in \
```



Methods

Participants grasped the handle of an IMT2 two degree of freedom robot (InMotion Technologies Inc.) as they reached from a start position to a movement target, located 20 cm away. The robot applied a velocity-dependent force to the hand during movement, according to eq. 1.

$$F_x = k\,[v_y] \tag{1}$$

In eq. 1, x and y are lateral and sagittal directions, F_x is the applied robot force in the left-right direction, v_y is hand velocity in the forward-backward direction and k=14 Ns/m.

Figure 3: robot2.pdf

```
-V mainfont=Helvetica \
-V monofont=Menlo \
-V fontsize=12pt \
-V colorlinks \
--latex-engine=xelatex \
--highlight-style=tango \
-o README.pdf
```

LaTeX

- LaTeX Font Catalogue

Markdown vs LaTeX

- Markdown has arguably nicer looking syntax
- Markdown has more output formats (via pandoc)
- LaTeX provides more control over fine-grained appearance

With pandoc you can take advantage of both. Provide pandoc with a LaTeX template, containing all of the formatting details you want, and keep the content in Markdown format.

Writing in Markdown

Using Pandoc to create a pdf

variables for LaTeX

Some examples:

Basic document creation with all of the LaTeX defaults:

`pandoc doc1.md -o doc1.pdf`

Let's make the sections and subsections numbered:

`pandoc doc1.md -o doc1.pdf -N`

Let's change the font to 12pt

`pandoc doc1.md -o doc1.pdf -N --variable fontsize=12pt`

Let's change the font to Palatino (the default LaTeX font is called Computer Modern):

`pandoc doc1.md -o doc1.pdf -N -V fontsize=12pt -V mainfont=Palatino --latex-engine=xelatex`

The page margins are to small for us, let's make them 1-inch all around:

```
pandoc doc1.md -o doc1.pdf -N -V fontsize=12pt -V
mainfont=Palatino --latex-engine=xelatex -V geometry:margin=1in
```

Where to put the formatting codes?

on the command line

pandoc doc1.md -o doc1.pdf -N -V fontsize=12pt -V mainfont=Palatino –latex-engine=xelatex -V geometry:margin=1in

Above we put various formatting codes (font, margins, etc) in the command-line command to run pandoc. There are other ways to do it.

in a YAML header in the main document

```
---
title: Hunting Replicants
author: Rick Deckard
date: November 1, 2019
papersize: letter
mainfont: "Myriad Pro"
fontsize: 12pt
geometry: margin=1.2in
---
```

Here we include, along with the title, author and date, various formatting options in the main document itself. Now to convert the plaintext document `ms.md` to pdf we can issue a simpler command:

```
pandoc ms.md -o ms.pdf --latex-engine=xelatex
```

We include the `--latex-engine=xelatex` option in the pandoc command, because we are asking for a nonstandard font. When converting to pdf, pandoc does this via LaTeX using the `pdflatex` conversion engine as a default. Unfortunately `pdflatex` doesn't handle non-default fonts particularly well (or at least, it's not very flexible). The `xelatex` LaTeX conversion engine handles fonts much more flexibly and allows you to use any font that's installed on your computer. Above I chose `Myriad Pro`.

in a separate file

We can put the formatting stuff into a separate file, called say `formatting.yaml`:

```
---
papersize: letter
mainfont: "Myriad Pro"
fontsize: 12pt
geometry: margin=1.2in
---
```

and leave the header of our main document for just the content, no formatting:

```
---
title: Hunting Replicants
author: Rick Deckard
date: November 1, 2019
---
```

The advantage of this is we really do separate out content from formatting. Now the command to convert from `ms.md` to pdf would be:

```
pandoc ms.md -o ms.pdf formatting.yaml --latex-engine=xelatex
```

We include `formatting.yaml` to tell pandoc to include the header info found in the file `formatting.yaml` when performing the conversion.