Academic Markdown

Who knew writing could be so nerdy?

Contents

About

Academic Markdown is a Python module that allows you generate `.html`, `.pdf`, `.docx`, and `.odt` files from Markdown source. Pandoc is used for most of the heavy lifting, so refer to the Pandoc website for detailed information about writing in Pandoc Markdown. However, Academic Markdown offers some additional functionality that is useful for writing scientific documents, such as integration with Zotero references, and a number of useful Academic Markdown extensions.

Download

You can download the latest release of Academic Markdown here:

- https://github.com/smathot/academicmarkdown/releases

Ubuntu users can install Academic Markdown from the Cogsci.nl PPA:

```
sudo add-apt-repository ppa:smathot/cogscinl
sudo apt-get update
sudo apt-get install python-academicmarkdown
```

Basic usage

Academic Markdown assumes that input files are encoded with `utf-8` encoding.

```
from academicmarkdown import build
build.HTML(u'input.md', u'output.html')
build.PDF(u'input.md', u'output.pdf')
build.DOCX(u'input.md', u'output.docx')
build.ODT(u'input.md', u'output.odt')
```

Dependencies

Academic Markdown has been tested exclusively on Ubuntu Linux. The following dependencies are required:

- pandoc is used for most of the heavy lifting. At the time of writing, the Ubuntu repositories do not contain a sufficiently recent version of Pandoc. Therefore, if you encounter trouble, try installing the latest version of Pandoc manually.
- pyzotero is necessary for extracting Zotero references.
- wkhtmltopdf is necessary for converting to `.pdf`. For best results, use the latest statically linked release, instead of the version from the Ubuntu repositories.

Zotero references

Since the basic Markdown conversion is performed by Pandoc, citations should be formatted as described on the Pandoc site:

- http://johnmacfarlane.net/pandoc/README.html#citations

You can automatically extract references from your Zotero library by setting the `zoteroApiKey` and `zoteroLibraryId` properties. Your references are not extracted from your local Zotero database, but through the web API of http://www.zotero.org. This means that you need to have a Zotero account and synchronize your local database with your account, in order to use this feature. You can find your your API key and library ID online on your Zotero profile page.

```
from academicmarkdown import build
build.zoteroApiKey = u'myapikey'
build.zoteroLibraryId = u'mylibraryid'
build.PDF(u'input.md', u'output.pdf')
```

Citations are split into separate terms using camelcase or undescore logic. An example of an underscore-style citation is `@bárány_halldén_1948`. And example of a camelcase-style citation is `@Land1999WhyAnimals`. Each citation is interpreted as a series of author names, followed by the year of publication, optionally followed by terms that match either the publication title, or the article title. So the following reference …

Land, M., Mennie, N., & Rusted, J. (1999). The roles of vision and eye movements in the control of activities of daily living. *Perception*, *28*(11), 1311–1328.

… matches any of the following terms:

- `@Land1999`
- `@Land1999Roles`
- `@LandMennie1999`
- `@LandRusted1999Percept`
- `@land_rusted_1999_percept`
- etc.

Note that you must consistently use the same citation to refer to a single reference in one document. If a citation matched multiple references from your Zotero database, one citation will be chosen at random.

Previous references will be cached automatically. To refresh, remove the file `.zoteromarkdown.cache` or run your Python script with the command-line argument: `--clear-cache`.

Academic Markdown extensions

Academic Markdown provides certain extensions to regular Markdown, in the form of YAML blocks embedded in `%-- --%` tags.

`figure`: figures

The `figure` block embeds a Figure in the text. Figures are numbered automatically. The ID can be used to refer to the Figure in the text, using a `%` character. So the following figure would be referred to as `%FigFA`.

```
%--
figure:
 id: FigFA
 source: example/foveal_acuity.png
 caption: "Visual acuity drops of rapidly with distance from the fovea"
--%
```

`exec`: external commands

The `exec` block inserts the return value of an external command in the text. For example, the following block embeds something like ‘Generated on 10/18/2013’:

```
%-- exec: "date +'Generated on %x'" --%
```

`include`: include other Markdown files

The `include` block includes an other Markdown file. For example:

```
%-- include: example/intro.md --%
```

`toc`: table of contents

The `toc` block will automatically generate a table of contents from the headings, assuming that headings are indicated using the `#` style and not the underlining style. You can indicate headings to be excluded from the table of contents as well.

```
%--
toc:
 mindepth: 1
 maxdepth: 2
 exclude: [Contents, Contact]
--%
```

Automatic DOI parsing

Applies only to `.html` and `.pdf`.

DOIs are automatically converted to http://dx.doi.org/ hyperlinks when they are prepended by `doi:`. For example, doi:10.3758/s13428-011-0168-7.

Page breaks

Applies only to `.html` and `.pdf`.

A single `~` preceded and followed by a blank line will automatically be converted to a page break.

Styling

You can specify a style by setting the `style` attribute:

```
from academicmarkdown import build
build.style = u'my-style'
build.PDF(u'input.md', u'output.pdf')
```

The `style` attribute must be a folder that contains a number of files (not all files need to exist):

- `html5.html` is the Pandoc HTML5 template used for generating HTML and PDF documents. See http://johnmacfarlane.net/pandoc/README.html#templates.
- `html5.css` is the Pandoc stylesheet used for generating HTML and PDF documents.
- `reference.csl` is a CSL citation style.
- `wkhtmltopdf.tmpl` is a shell command that is used to convert HTML to PDF. It can have something like the following contents:
  `wkhtmltopdf -L 20 -R 20 "%(source)s" "%(target)s"`

License