Pandoc/Defaults

Carsten Allefeld

2022–1–12

Contents

This document describes a way to define "formats" (Pandoc processing instructions) and associate Markdown files with them, and specifies the behavior of a program which processes such files.

1 Background

1.1 Motivation

Pandoc is an extremely versatile tool for document conversion, centered on its own feature-rich version of Markdown which makes it possible to write complex documents. There are many purposes Pandoc can be used for, and a large number of command line options allow to tailor Pandoc's functions to these special uses.

For many people, Pandoc has become an integral part of their daily routine, much like Microsoft Word and PowerPoint are used by others. In this context, there are typically a small number of use cases which repeat over and over again, and specifying command line options for them manually is tedious and error-prone. Such use cases might be:

- create a slide presentation (with specific color scheme and logo),

- write a technical report,
- write a letter (with a specific letterhead),
- prepare an academic publication,
- or to simply collect some notes on a project.

In the following I will call the specific configuration for a repeated use case a “format”.

There have been a number of attempts to wrap Pandoc in a way that makes it easy to store and re-use such formats:

- panzer, which “adds *styles* to pandoc”,
- rmarkdown, an R package which binds together knitr processing of Markdown-with-code with a number of output formats defined in R,
- pandocomatic, with which “you can express common patterns of using pandoc for generating your documents”,
- panrun, a “minimal script that runs pandoc with the options it finds in the YAML metadata of the input markdown file”,
- Pandoc/PDF, a package for the Atom editor which allows to associate file extensions with Pandoc options,
- and probably several others.

The problem is that there is no common standard for defining such formats, and for associating them with a Markdown file. (Yes, I am aware of the “Standards” xkcd, but see next section.) Moreover, the route taken by some of these packages, to specify options in detail within the file, goes against the idea of resuable formats, and tends to reproduce the abominably endless preambles of LaTeX files as well as the variety of slightly differently formatted Word documents.

1.2 Pandoc’s defaults files as formats

Since version 2.8 (and reasonably bug-free since 2.9.1), Pandoc supports a command line argument `--defaults` with which a so-called “defaults file” is specified. This file is in YAML format and makes it possible to configure all aspects of the Pandoc conversion process in a single file. With this, there is now a “common standard” for defining formats endorsed by the creator of Pandoc himself.

With this option, running Pandoc in a customized way becomes as simple as e.g.

```
pandoc --defaults=slides_beamer.yaml slides.md \
    --output=slides.pdf
```

where `slides_beamer.yaml` is a defaults file which specifies in detail how the input document is to be converted to PDF slides (here via LaTeX using the Beamer document class). Having another defaults file `slides_powerpoint.yaml`, the same input file could be converted into a PowerPoint `pptx`-file by:

```
pandoc --defaults=slides_powerpoint.yaml slides.md \
    --output=slides.pptx
```

This opens up the possibility to create a simple user interface to Pandoc supporting different kinds of documents and styles. The user creates one or more defaults files, and for a given conversion only has to specify two pieces of information:

- the name of the defaults file
- and the file extension of the output.

Note that in the first example above, the extension `pdf` of the output file was necessary to trigger the creation of a PDF. Any other extension would have let the Beamer-LaTeX code to be written to the file, which means only the extensions `pdf` and `tex` make sense in combination with this defaults file. In the second example, only the extension `pptx` makes sense.

For the use cases listed above, defaults files could be:

- `slides_beamer.yaml`, `slides_powerpoint.yaml`
- `report_html.yaml`, `report_word.yaml`
- `letter_word.yaml`
- `paper_latex.yaml`
- `notes_html.yaml`

1.3 Associating a Markdown file with formats

What is missing is a simple way to specify these two pieces of information, apart from writing them on the command line. This is especially important if the conversion process is to be started from an editor, where specifying options is cumbersome or requires complex configuration options.

There are several possible strategies, but the simplest seems to be to add a key to the initial YAML block of a Markdown file, within which output file extensions and defaults files are associated:

```
pandoc-defaults_:
    - pdf: slides_beamer
    - pptx: slides_powerpoint
```

2 Specification

A Pandoc/Defaults processor is a command-line program or a function within an editor program, e.g. triggered by a keyboard shortcut.

If it is invoked upon a Markdown file `NAME.EXT` with an initial YAML block which includes the top-level key `pandoc-defaults_`, it expects the associated value to be a sequence of single-key mappings, where keys and values are string scalars. Keys have to be unique within the sequence. Keys are interpreted as output file extensions and values as names of Pandoc defaults files (without the extension `.yaml`):

```
---
…
pandoc-defaults_:
    - EXT1: DEFAULTS1
    - EXT2: DEFAULTS2
    …
…
...
```

It is then to spawn, in the directory containing the file, one Pandoc process for each key/value pair:

```
pandoc --defaults=DEFAULTS1.yaml NAME.EXT --output=NAME.EXT1
pandoc --defaults=DEFAULTS2.yaml NAME.EXT --output=NAME.EXT2
…
```

As a result, files with the specified extensions are created in the same directory as the Markdown file, each using the format defined by the associated defaults file. Additional Pandoc arguments may be included, as long as they do not substantially affect the result, e.g. `--verbose`.

To simplify the use of *default* defaults files, if the value for a key is omitted (equivalent to YAML `null` or `~`), it is to be set identical to the key. That means the following two entries are equivalent:

```
pandoc-defaults_:
    - html:
    - docx:
```

```
pandoc-defaults_:
    - html: html
    - docx: docx
```

To accomodate the case of simple one-off Markdown files, either without an initial YAML block or without a top-level key `pandoc-defaults_`, in this case the Pandoc/Defaults processor is to act as if the document contained the entry

```
pandoc-defaults_:
    - html:
```

Referencing a defaults file, either explicitly through a value or implicitly by omitting it, assumes that such a file can be found by Pandoc. That means that the defaults file has to be either installed in the `defaults` subdirectory of the Pandoc user data directory, or that it has to be present in the directory of the Markdown file.

If the user wishes to use Pandoc's built-in settings and templates unchanged, they can use a minimal defaults file. For example, `html.yaml` could simply contain the lines:

```
writer: html5
standalone: true
```

The behavior described above is the core functionality expected from a Pandoc/Defaults processor, but it may offer further functions. For example, an editor may

- provide another keyboard shortcut, in response to which only the first key/value pair is used,
- provide a drop-down menu with entries corresponding to the key/value pairs, so that in response to a selection only the selected key/value pair is used, or
- automatically open the file created from the first key/value pair in a viewer application or tab.

If the file extension suggests preprocessing, a Pandoc/Defaults processor may run the preprocessor before invoking Pandoc on the resulting file; examples are `Rmd` files to be processed with knitr or `pmd` files to be processed with Pweave on files with the extension `pmd`.

3 Design principles

- Pandoc/Defaults is designed to rely as much as possible on existing functionality of Pandoc, adding only what cannot be achieved using Pandoc alone.
- Format information is added to the Markdown file itself in order to avoid complex configuration, e.g. based on file extensions or parts of the file name, or to have directories cluttered with sidecar configuration files. Note however that this system is compatible with using formats defined locally but separately, due to the way Pandoc searches for defaults files.
- The YAML key under which formats are defined ends in an underscore so that it does not affect processing by Pandoc.
- Formats are listed in a sequence of single-key mappings (the standard way to represent an ordered mapping in YAML), because a simple YAML mapping has no defined order; this way it is possible to identify a “first” format.
- Output file extensions are used as keys because it does not make sense to create more than one file with the same name.
- Pandoc is to be run in the directory of the input file so that local external resources (images, defaults files) are found.