An Introduction to rmdplugr

Johan Larsson

Contents

1 Motivation

The motivation for **rmdplugr** was first off to produce a cleaner latex output format for rmarkdown documents, given that the default settings modify standard latex template by introducing

- wide margins,
- compact titles, and
- paragraph spacing (instead of indentation).

All of these make changes to the, in my opinion, well-founded design choices and should be alternatives rather than the default. I understand the stance taken by the pandoc team that output should be made similar across different outputs, yet I believe that the vast majority of users—particularly of Rmarkdown—make use only of a single output format at a time.

As a secondary objective, I wanted to allow modifications of the standard latex template to allow for things such as headers and footers and author affiliations.

2 rmdplugr

I did not, however, want to produce *yet another* custom latex template needing to be continuously updated along with the default templates, nor having to cater to different versions of pandoc, having to keep separate templates for different versions (as rmarkdown does).

rmdplugr was created to avoid such issues and does so by instead modifying the default latex template in a version-agnostic fashion. The default template is pulled by calling

```
pandoc -D latex
```

The template is then modified with modifications in the form of *plugins*, which are merged into the latex template—sometimes replacing other material, but usually inserting new material, such as support for footers and headers.

In this way, rmdplugr doesn't have to adapt to the pandoc version and can even be used with custom templates *provided* that some of the matter from the default template is left intact.

These *plugins* can be turned on and off via the `output` setting in the metadata block and their options are controlled by metadata settings.

3 Plugins

3.1 `pdf_article()`

3.1.1 Marginal material

Headers and footers are provided using **fancyhdr** for standard latex document classes and **scrlayer-page** for KOMA-script classes. Simply adding

```
header: true
```

in the metadata block will invoke the default settings using the `automark` option in **scrlayer-page** and the default `\pagestyle{fancy}` for **fancyhdr**. More advanced settings can be provided by using

```
header:
  l: Johan Larsson
  c: Document title
footer:
  r: Date
```

3.1.2 Author blocks

Support for author blocks is made available through the **authblk** package if `author_block = TRUE` in the call to `pdf_article()`. This module uses the usual `author` metadata along with `affiliation`. There are multiple ways to use this plugin. If you are a single author you can simply do

```
author: Johan Larsson
affiliation:
  - Lund University
  - Another affiliation
```

and you'll get an authors block. If you have a more complicated list of authors, you can use the following format, where authors are coupled to affiliations via the `id` tag.

```yaml
author:
  - name: Johan Larsson
    id: "1,2"
  - name: Another Author
    id: 1
affiliation:
  - name: Lund University
    id: 1
  - name: Another affiliation
    id: 2
```

The type of author blocks (and whatever other options are available to the **authblk** package) can be set using the metadata option `author-options`. Moreover, the default settings for fonts of affiliations and authors can be set using `affilfont` and `authfont` respectively.

3.1.3 No indentation

The default in `pdf_article()` is to use indentations rather than paragraph spacing. The trouble with this, however, is that some of the latex environments will always be surrounded by empty lines no matter how the text is formatted in the markdown document; this means that the paragraph following such environments will always be indented, which is not always wanted.

To circumvent this, the `noindentafter` plugin loads the **noindentafter** package, which makes sure that there is to be no indentation after

- itemize,
- enumerate,
- description, and
- code block

environments. You can in fact see the result of this for the list above, which would otherwise have required a `\noindent` command before this paragraph.

3.1.4 Subfigs

The `subfigs` plugin is a very small plugin that simply makes sure that the **subfigs** package is loaded. It is needed for subfigs in rmarkdown documents.

3.2 `pdf_presentation()`

This output format is based on the `rmarkdown::beamer_presentation()` format. It adds a couple of plugins.

3.2.1 Frame numbering

To enable frame numbering for your beamer presentations, setting `frame_numbering = TRUE` will invoke simple frame numbers at the bottom-right of each slide.

3.2.2 Subfigs

See section 3.1.4.

3.2.3 More font themes

In `beamer_presentation()` you can only choose a single font theme for beamer. This simple plugin allows a list of fontthemes, so that you can do

```
fonttheme:
  - structurebold
  - professionalfonts
```

which is sometimes useful.