A Pandoc Markdown Article Starter and Template

true

`r format(Sys.time(), '%B %d, %Y')`

Abstract

This document provides an introduction to R Markdown, argues for its benefits, and presents a sample manuscript template intended for an academic audience. I include basic syntax to R Markdown and a minimal working example of how the analysis itself can be conducted within R with the `knitr` package.

Introduction

Academic workflow, certainly in political science, is at a crossroads. The *American Journal of Political Science* (*AJPS*) announced a (my words) "show your work" initiative in which authors who are tentatively accepted for publication at the journal must hand over the raw code and data that produced the results shown in the manuscript. The editorial team at *AJPS* then reproduces the code from the manuscript. Pending successful replication, the manuscript moves toward publication. The *AJPS* might be at the fore of this movement, and it could be the most aggressive among political science journals, but other journals in our field have signed the joint Data Access & Research Transparency (DART) initiative. This, at a bare minimum, requires uploading code from quantitatively-oriented published articles to in-house directories hosted by the journal or to services like Dataverse.

There are workflow implications to the Lacour controversy as well. Political science, for the foreseeable future, will struggle with the extent of the data fraud perpetrated by Michael Lacour in an article co-authored with Donald P. Green in *Science*, the general scientific journal of record in the United States. A failure to reproduce LaCour's results with different samples uncovered a comprehensive effort by LaCour to "fake" data that provided results to what we felt or believed to be true (i.e. "truthiness"). However, fake data can have real consequences for both the researcher and those who want to learn from it and use it for various purposes. Even research done honestly may suffer the same fate if researchers are not diligent in their workflow.

These recent events underscore the DART push and cast a shadow over our workflow. However, good workflow has always been an issue in our discipline. Cloud storage services like Dropbox are still relatively new among political scientists. Without cloud storage, previous workflow left open the possibility that work between a home computer and an office computer was lost as a function of a corrupted thumb drive, an overheated power supply, or, among other things, the wave of viruses that would particularly affect Microsoft users every summer. Social sciences, unlike engineering, have traditionally relied on software like Microsoft Word for manuscript preparation though any word processor reduces workflow to a series of clicks and strokes on a keyboard. This is a terrible way to track changes or maintain version control. The addition of collaborators only compounds all the aforementioned issues. The proverbial left hand may not know what the right hand is doing.

I think there is reason for optimism. We only struggle with it now because we have tools like R Markdown and Pandoc, more generally, that make significant strides in workflow. LaTeX resolved earlier issues of corrupted binary files by reducing documents to raw markup that was little more than raw text and revisions that could be easily kept as "commented" text. However, for all its benefits (including pretty PDFs), LaTeX is *ugly* code and does not provide means of seamlessly working with the actual data analysis itself. R Markdown both eliminates markup and allows the author and her collaborators to write and reproduce the manuscript in one fell swoop.

Getting Started with YAML

The lion's share of a R Markdown document will be raw text, though the front matter may be the most important part of the document. R Markdown uses YAML for its metadata and the fields differ from what an author would use for a Beamer presentation. I provide a sample YAML metadata largely taken from this exact document and explain it below.

"'{r eval=FALSE}

---

output:
pdf_document:
citation_package: natbib
keep_tex: true
fig_caption: true
latex_engine: pdflatex
template: ~/Dropbox/miscelanea/svm-r-markdown-templates/svm.latex.ms.tex
title: "A Pandoc Markdown Article Starter and Template"
thanks: "Replication files are available on the author's Github account…"
author:
- name: Steven V. Miller
affiliation: Clemson University
- name: Mary Margaret Albright
affiliation: Pendelton State University
- name: Rembrandt Q. Einstein
affiliation: Springfield University
abstract: "This document provides an introduction to R Markdown, argues for its…"
keywords: "pandoc, r markdown, knitr"
date: "`r format(Sys.time(), '%B %d, %Y')`"
geometry: margin=1in
fontfamily: mathpazo
fontsize: 11pt
spacing: double
bibliography: ~/Dropbox/master.bib
biblio-style: apsr

"'

`output:` will tell R Markdown we want a PDF document rendered with LaTeX. Since we are adding a fair bit of custom options to this call, we specify `pdf_document:` on the next line (with, importantly, a two-

space indent). We specify additional output-level options underneath it, each are indented with four spaces. `citation_package: natbib` tells R Markdown to use `natbib` to handle bibliographic citations.[1] Thereafter, the next line (`keep_tex: true`) tells R Markdown to render a raw `.tex` file along with the PDF document. This is useful for both debugging and the publication stage, when the editorial team will ask for the raw `.tex` so that they could render it and later provide page proofs. The next line `fig_caption: true` tells R Markdown to make sure that whatever images are included in the document are treated as figures in which our caption in brackets in a Markdown call is treated as the caption in the figure. The next line (`latex_engine: pdflatex`) tells R Markdown to use pdflatex and not some other option like `lualatex`. For my template, I'm pretty sure this is mandatory.[2]

The next line (`template: ...`) tells R Markdown to use my custom LaTeX template.[3] While I will own any errors in the code, I confess to "Frankensteining" this template from the default LaTeX template from Pandoc, Kieran Healy's LaTeX template, and liberally using raw TeX from the Association for Computing Machinery's (ACM) LaTeX template. I rather like that template since it resembles standard manuscripts when they are published in some of our more prominent journals. I will continue with a description of the YAML metadata in the next paragraph, though invite the curious reader to scroll to the end of the accompanying post to see the PDF this template produces.

The next fields get to the heart of the document itself. `title:` is, intuitively, the title of the manuscript. Do note that fields like `title:` do not have to be in quotation marks, but must be in quotation marks if the title of the document includes a colon. That said, the only reason to use a colon in an article title is if it is followed by a subtitle, hence the optional field (`subtitle:`). Notice I "comment out" the subtitle in the above example with a pound sign since this particular document does not have a subtitle. If `thanks:` is included and has an accompanying entry, the ensuing title of the document gets an asterisk and a footnote. This field is typically used to advise readers that the document is a working paper or is forthcoming in a journal.

The next field (`author:`) is a divergence from standard YAML, but I think it is useful. I will also confess to pilfering this idea from Kieran Healy's template. Typically, multiple authors for a given document are separated by an `\and` in this field. However, standard LaTeX then creates a tabular field separating multiple authors that is somewhat restrictive and not easy to override. As a result, I use this setup (again, taken from Kieran Healy) to sidestep the restrictive rendering of authors in the standard `\maketitle` tag. After `author:`, enter `- name:` (no space before the dash) and fill in the field with the first author. On the next line, enter two spaces, followed by `affiliation:` and the institute or university affiliation of the first author.

Do notice this can be repeated for however many co-authors there are to a manuscript. The rendered PDF will enter each co-author in a new line in a manner similar to journals like *American Journal of Political Science*, *American Political Science Review*, or *Journal of Politics*.

The next two fields pertain to the frontmatter of a manuscript. They should also be intuitive for the reader. `abstract` should contain the abstract and `keywords` should contain some keywords that describe the research project. Both fields are optional, though are practically mandatory. Every manuscript requires an abstract and some journals—especially those published by Sage—request them with submitted manuscripts. My template also includes these keywords in the PDF's metadata.

[1] R Markdown can use Pandoc's native bibliography management system or even `biblatex`, but I've found that it chokes with some of the more advanced stuff I've done with my .bib file over the years. For example, I've been diligent about special characters (e.g. umlauts and acute accents) in author names in my .bib file, but Pandoc's native citation system will choke on these characters in a .bib file. I effectively need `natbib` for my own projects.

[2] The main reason I still use `pdflatex` (and most readers probably do as well) is because of LaTeX fonts. Unlike others, I find standard LaTeX fonts to be appealing.

[3] Notice that the path is relative. The user can, if she wishes, install this in the default Pandoc directory. I don't think this is necessary. Just be mindful of wherever the template is placed. Importantly, `~` is used in R to find the home directory (not necessarily the working directory). It is equivalent to saying `/home/steve` in Linux, or `/Users/steve` on a Mac, in my case.

`date` comes standard with R Markdown and you can use it to enter the date of the most recent compile. I typically include the date of the last compile for a working paper in the `thanks:` field, so this field currently does not do anything in my Markdown-LaTeX manuscript template. I include it in my YAML as a legacy, basically.

The next items are optional and cosmetic. `geometry:` is a standard option in LaTeX. I set the margins at one inch, and you probably should too. `fontfamily:` is optional, but I use it to specify the Palatino font. The default option is Computer Modern Roman. `fontsize:` sets, intuitively, the font size. The default is 10-point, but I prefer 11-point. `spacing:` is an optional field. If it is set as "double", the ensuing document is double-spaced. "single" is the only other valid entry for this field, though not including the entry in the YAML metadata amounts to singlespacing the document by default. Notice I have this "commented out" in the example code.

The final two options pertain to the bibliography. `bibliography:` specifies the location of the .bib file, so the author could make citations in the manuscript. `biblio-style` specifies the type of bibliography to use. You'll typically set this as APSR. You could also specify the relative path of my *Journal of Peace Research* .bst file if you are submitting to that journal.

Getting Started with Markdown Syntax

There are a lot of cheatsheets and reference guides for Markdown (e.g. Adam Prichard, Assemble, Rstudio, Rstudio again, Scott Boms, Daring Fireball, among, I'm sure, several others). I encourage the reader to look at those, though I will retread these references here with a minimal working example below.

```
# Introduction

**Lorem ipsum** dolor *sit amet*.

- Single asterisks italicize text *like this*.
- Double asterisks embolden text **like this**.

Start a new paragraph with a blank line separating paragraphs.

- This will start an unordered list environment, and this will be the first item.
- This will be a second item.
- A third item.
    - Four spaces and a dash create a sublist and this item in it.
- The fourth item.

1. This starts a numerical list.
2. This is no. 2 in the numerical list.

# This Starts A New Section
## This is a Subsection
### This is a Subsubsection
#### This starts a Paragraph Block.
```

```
> This will create a block quote, if you want one.

Want a table? This will create one.

Table Header  | Second Header
------------- | -------------
Table Cell    | Cell 2
Cell 3        | Cell 4

Note that the separators *do not* have to be aligned.

Want an image? This will do it.

![caption for my image](path/to/image.jpg)

`fig_caption: yes` will provide a caption. Put that in the YAML metadata.

Almost forgot about creating a footnote.[^1] This will do it again.[^2]

[^1]: The first footnote
[^2]: The second footnote

Want to cite something?

- Find your biblatexkey in your bib file.
- Put an @ before it, like @smith1984, or whatever it is.
- @smith1984 creates an in-text citation (e.g. Smith (1984) says...)
- [@smith1984] creates a parenthetical citation (Smith, 1984)

That'll also automatically create a reference list at the end of the document.

[In-text link to Google](http://google.com) as well.
```

That's honestly it. Markdown takes the chore of markup from your manuscript (hence: "Markdown").

On that note, you could easily pass most LaTeX code through Markdown if you're writing a LaTeX document. However, you don't need to do this (unless you're using the math environment) and probably shouldn't anyway if you intend to share your document in HTML as well.

Using R Markdown with Knitr

Perhaps the greatest intrigue of R Markdown comes with the `knitr` package provided by @xie2013ddrk. In other words, the author can, if she chooses, do the analysis in the Markdown document itself and compile/execute it in R.

Take, for example, this simple exercise using the `voteincome` data from the `Zelig` package. Suppose I want

to explain the decision to vote using data from this package. I load in the data, clean the data, run the analyses, and present the results as a coefficient plot.

Here's what this code looks like. All I did was create a code display, which starts with three *backticks* (i.e. those ticks next to the number 1 key on your keyboard) and ends with three backticks on another line. On the first line of backticks (i.e. to start the code display) enter `{r, eval=FALSE, tidy=TRUE}`. The `eval=FALSE` option just displays the R code (and does not run it), `tidy=TRUE` wraps long code so it does not run off the page.

Within that code display, I enter my R code like this.

“‘{r, eval=FALSE, tidy = TRUE} suppressMessages(library(Zelig)) suppressMessages(library(arm)) suppressMessages(library(coefplot))

data(voteincome)

voteincome$z.age <- arm::rescale(voteincome$age)$ voteincome$z.education <- arm::rescale(voteincome$education)$ voteincome$z.income <- arm::rescale(voteincome$income)$

M1 <- glm(vote ~ z.age + female + z.education + z.income, data=voteincome, family=binomial)

coefplot(M1) “‘

The implications for workflow are faily substantial. Authors can rather quickly display the code they used to run the analyses in the document itself (likely in the appendix). As such, there's little guesswork for reviewers and editors in understanding what the author did in the analyses reported in the manuscript.

It doesn't end there. In fact, here's what happens when `eval=FALSE` is omitted or changed to `eval=TRUE`. Now, the code runs within R. Observe.

“‘{r, eval=TRUE, tidy = TRUE, cache=FALSE, fig.cap=“A Coefficient Plot”} suppressMessages(library(Zelig)) suppressMessages(library(arm))

data(voteincome)

voteincome$z.age <- arm::rescale(voteincome$age)$ voteincome$z.education <- arm::rescale(voteincome$education)$ voteincome$z.income <- arm::rescale(voteincome$income)$

M1 <- glm(vote ~ z.age + female + z.education + z.income, data=voteincome, family=binomial)

arm::coefplot(M1) “‘

To get `knitr` to present the results of a table, add `results="asis"` to the brackets to start the R code chunk. The ensuing output will look like this (though the table may come on the next page).

“‘{r, eval=TRUE, tidy = TRUE, size=“small”, cache=FALSE, results=“asis”} suppressMessages(library(Zelig)) suppressMessages(library(stargazer)) suppressMessages(library(arm))

data(voteincome)

voteincome$z.age <- arm::rescale(voteincome$age)$ voteincome$z.education <- arm::rescale(voteincome$education)$ voteincome$z.income <- arm::rescale(voteincome$income)$

M1 <- glm(vote ~ z.age + female + z.education + z.income, data=voteincome, family=binomial)

stargazer(M1, title=“A Handsome Table”, header=FALSE) “‘

Adding `echo="FALSE"` inside the brackets to start the R chunk will omit the presentation of the R commands. It will just present the table. This provides substantial opportunity for authors in doing their analyses. Now, the analysis and presentation in the form of a polished manuscript can be effectively simultaneous.[4]

[4] I'm not sure if I'm ready to commit to this myself since my workflow is still largely derived from Rob J. Hyndman's example. However, *knitr* has endless potential, especially when analyses can stored in cache, saved as chunks, or loaded in the preamble of a document to reference later in the manuscript.