R Markdown for Psychology Graduate Students

Holly Zaharchuk

2020-05-04

Contents

Chapter 1

Welcome

I designed this guide to be a resource for psychology graduate students looking to streamline their research pipelines. With R Markdown, you can load, clean, manipulate, analyze, and present your data in one environment. This guide focuses on the presentation piece, with information on creating slides, posters, manuscripts, CVs, and reports in several formats, including HTML, PDF, and Microsoft Word.

1.1 Background

This guide assumes a basic level of familiarity with R and RStudio. If you don't know how to use either of these, there are several beginner tutorials you should check out first. I have also created primers for Beginner and Intermediate R in Chapter **??**.

1.2 Getting started

What is R Markdown?

Markdown is a specific markup language with plain text-formatting syntax. R Markdown is a specific markdown variety.

R Markdown and R are not the same thing. R Markdown combines R code (or code from other programming languages) and markdown in the RStudio integrated development environment (IDE). This allows you to embed code and text in the same document.

You can install the `rmarkdown` package from CRAN or GitHub:

```
install.packages("rmarkdown")
# or the development version:
# devtools::install_github("rstudio/rmarkdown")
```

1.2.1 Tips

- Treat your data as read-only
- Comment code early and often
- Keep code chunks small
- Label chunks to help with diagnosing issues
- Nest all files under one directory (if possible)

1.2.2 General reference documents

- R Markdown Guide
- R Markdown Cheat Sheet
- R Markdown Reference Guide
- Keyboard shortcuts
- `knitr` documentation

Chapter 2

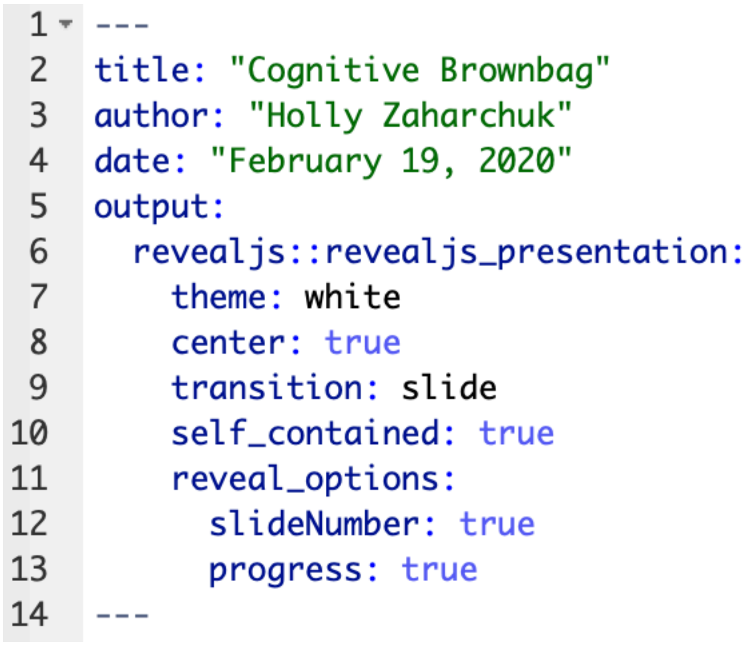
Parts of a document

1. YAML header
2. Markdown
3. Code chunks

2.1 YAML header

The first part of your document is called the YAML header. This is where you set the global options for the output and formatting. The YAML header appears at the top of your document, and is defined by three dashes/hyphens at the beginning and end.

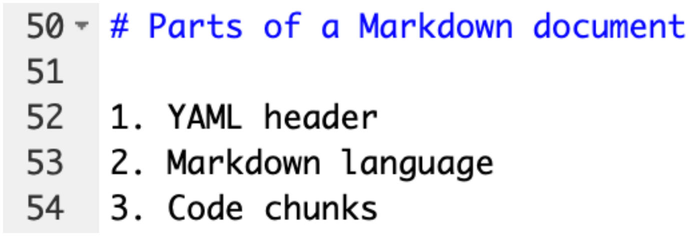
In the example below, I show the YAML header for a set of `revealjs` slides. I've included basic information, like the title and date, in addition to template-specific parameters, like whether there should be slide numbers or not. Section **??** has more information on setting YAML formatting parameters.



```
---
title: "Cognitive Brownbag"
author: "Holly Zaharchuk"
date: "February 19, 2020"
output:
  revealjs::revealjs_presentation:
    theme: white
    center: true
    transition: slide
    self_contained: true
    reveal_options:
      slideNumber: true
      progress: true
---
```

2.2 Markdown

The plain text-formatting syntax of R Markdown allows for conversion to multiple document types. The image below shows an example of the basic syntax. The # denotes a header, while the numbered list behaves like you would expect one to in Word. However, unlike Word, the actual numbers don't matter; I could've put all 1's here, and R Markdown would've formatted them for me. Go to Chapter **??** for more information on markdown syntax.



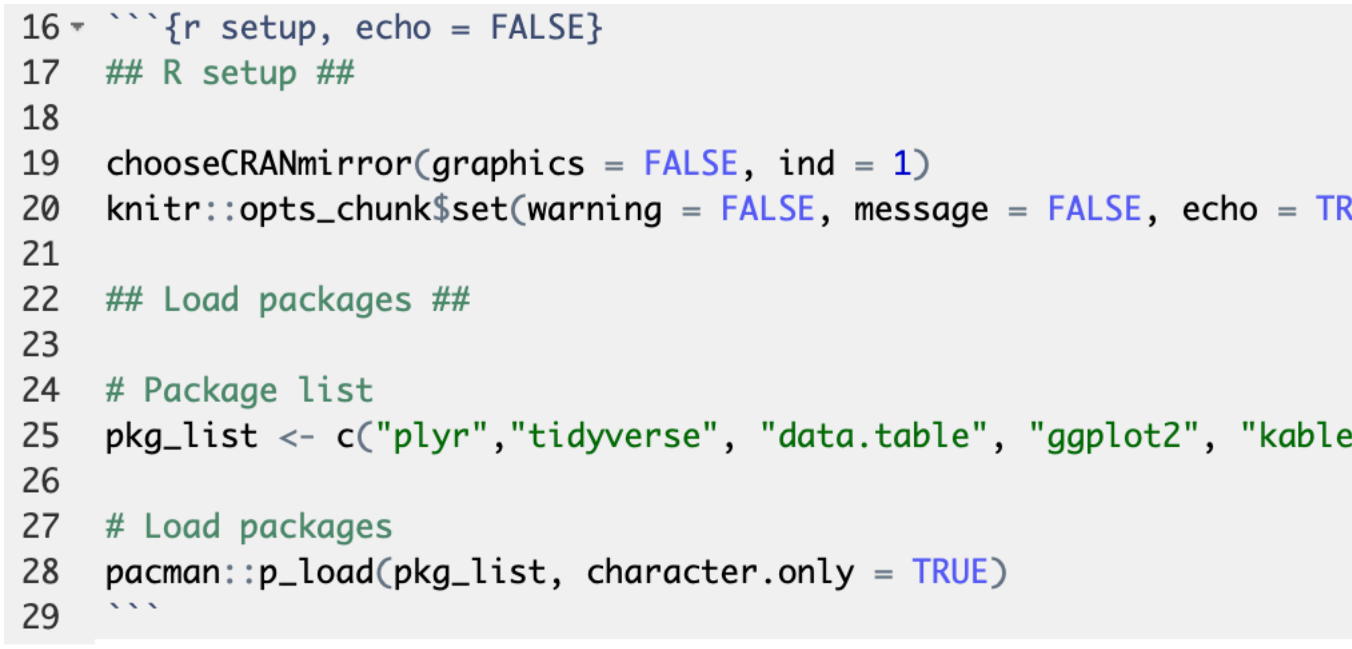
```
# Parts of a Markdown document

1. YAML header
2. Markdown language
3. Code chunks
```

2.3 Code chunks

Code chunks are one of the core features of R Markdown. Code chunks are set apart from markdown by three backticks at the beginning and end. In curly brackets after the first set of backticks, you specify the coding language you want to use (here, it's R, with a lowercase r).

You can also add other arguments, like a name for the chunk (here, it's `setup`), and specific chunk options. The example I've provided below is the first chunk in my R Markdown document. It establishes the default chunk options with `knitr::opts_chunk$set`. You can see that while I've set `echo = TRUE` globally, so that all of my code chunks appear in the document by default, I set `echo = FALSE` for this specific chunk. A full list of chunk options can be found here.

```
# This is a chunk of R code that adds an image
knitr::include_graphics("images/example_chunk.png")
```



~~~
```{r setup, echo = FALSE}
## R setup ##

chooseCRANmirror(graphics = FALSE, ind = 1)
knitr::opts_chunk$set(warning = FALSE, message = FALSE, echo = TR

## Load packages ##

# Package list
pkg_list <- c("plyr","tidyverse", "data.table", "ggplot2", "kable

# Load packages
pacman::p_load(pkg_list, character.only = TRUE)
```
~~~

There are multiple ways to run code chunks to test them in RStudio before creating your output. You can run code like you would in R by highlighting the relevant lines of code and hitting CTRL/command + enter. You can also hit the green "play" button in the upper right-hand corner of the chunk.

Each chunk is an island, so if you haven't run a previous chunk that contains some variable you need in the chunk you want to run, it'll throw an error. At the top right of your open .Rmd document in RStudio, you'll also see a **Run** dropdown menu. There, you can choose different options for running certain code chunks.

Tip: you can use the chunk option "cache = TRUE" for very time-consuming chunks, but there are some catches as described here.

Chapter 3

Outputs

R Markdown can transform plain text and code into several different document formats. There are also multiple ways to **Knit** or `render` the output.

3.1 Output options

3.1.1 html_document

HTML is overall the most flexible. It supports the types of content we're interested in creating as graduate students—tables, graphs, and the like—and you can easily transform your HTML output to a PDF with `pagedown::chrome_print(file)` if you have Google Chrome. The poster template I use follows this process.

3.1.2 pdf_document

If you're familiar with LaTeX, you may be inclined to output directly to a PDF, since you can include inline LaTeX code in your documents (more on this in Chapter **??**). The CV and manuscript templates that I use rely on the fine-grained typsetting capabilities of LaTeX. To create a PDF, you need to have LaTeX installed locally. If you don't already, you can install a `tinytex` distribution through the R console.

```r
install.packages("tinytex")
```

3.1.3 word_document

You can also output to Microsoft Word and Powerpoint. I often work with colleagues who prefer to edit in Word, so sometimes I need to do this. I prefer not to if I can help it though, because you lose several important functions. For