papaja: Reproducible APA manuscripts with R Markdown

Frederik Aust

2017-10-28

Contents

Preface

This book will be a manual for the R package papaja. It is currently in the process of being written; I'm publishing new sections of the book as I'm writing them. If you have suggestions for improvements or additional topics you would like to see covered, please let me know by opening an issue on GitHub or creating a pull request.

`papaja` has not yet been submitted to CRAN because it is under active development. Currently, there are still a couple of loose ends we would like to tie up before we release the package to a larger audience. If you would like to contribute to speed up the process, have a look at the chapters Limitations and Future directions.

If you are interested in reproducible research with R more generally, I recommend the books by **?** and **?**. Furthermore, **?** give an overview of tools, practices, and plattforms to implement reproducible research.

This manual is fully reproducible and was written in R [3.4.2, **?**] and the R-packages *afex* [0.18.0, **?**], *bindrcpp* [0.2, **?**], *citr* [0.2.0, **?**], *DiagrammeR* [0.9.2, **?**], *dplyr* [0.7.4, **?**], *knitr* [1.17, **?**], *lme4* [1.1.14, **?**], *lsmeans* [2.27.2, **?**], *papaja* [0.1.0.9492, **?**], *shiny* [1.0.5, **?**], and *wordcountaddin* [0.2.0, **?**].

```
## Session info -------------------------------------------------------------
##  setting  value
##  version  R version 3.4.2 (2017-09-28)
##  system   x86_64, linux-gnu
##  ui       X11
##  language de_DE
##  collate  de_DE.UTF-8
##  tz       Europe/Berlin
##  date     2017-10-28
## Packages -----------------------------------------------------------------
##  package      * version    date       source
##  assertthat     0.2.0      2017-04-11 CRAN (R 3.4.0)
##  backports      1.1.1      2017-09-25 CRAN (R 3.4.2)
##  base         * 3.4.2      2017-09-29 local
##  BayesFactor    0.9.12-2   2015-09-19 CRAN (R 3.3.3)
##  bindr          0.1        2016-11-13 CRAN (R 3.4.0)
##  bindrcpp       0.2        2017-06-17 CRAN (R 3.4.0)
##  bookdown       0.5        2017-08-20 cran (@0.5)
##  brew           1.0-6      2011-04-13 CRAN (R 3.3.3)
##  coda           0.19-1     2016-12-08 CRAN (R 3.3.3)
##  colorspace     1.3-2      2016-12-14 CRAN (R 3.3.3)
##  compiler       3.4.2      2017-09-29 local
##  datasets     * 3.4.2      2017-09-29 local
##  devtools       1.13.3     2017-08-02 CRAN (R 3.4.2)
##  DiagrammeR   * 0.9.2      2017-09-06 CRAN (R 3.4.1)
##  digest         0.6.12     2017-01-27 CRAN (R 3.3.3)
##  downloader     0.4        2015-07-09 CRAN (R 3.4.1)
##  dplyr          0.7.4      2017-09-28 CRAN (R 3.4.2)
##  evaluate       0.10.1     2017-06-24 CRAN (R 3.4.2)
##  ggplot2        2.2.1      2016-12-30 CRAN (R 3.3.3)
##  glue           1.1.1      2017-06-21 CRAN (R 3.4.0)
##  graphics     * 3.4.2      2017-09-29 local
##  grDevices    * 3.4.2      2017-09-29 local
##  grid           3.4.2      2017-09-29 local
##  gridExtra      2.3        2017-09-09 CRAN (R 3.4.1)
##  gtable         0.2.0      2016-02-26 CRAN (R 3.3.3)
##  gtools         3.5.0      2015-05-29 CRAN (R 3.3.3)
##  hms            0.3        2016-11-22 CRAN (R 3.4.1)
##  htmltools      0.3.6      2017-04-28 CRAN (R 3.4.2)
##  htmlwidgets    0.9        2017-07-10 CRAN (R 3.4.1)
##  igraph         1.1.2      2017-07-21 CRAN (R 3.4.1)
##  influenceR     0.1.0      2015-09-03 CRAN (R 3.3.3)
##  jsonlite       1.5        2017-06-01 cran (@1.5)
##  knitr          1.17       2017-08-10 CRAN (R 3.4.2)
##  latex2exp      0.4.0      2015-11-30 CRAN (R 3.4.1)
##  lattice        0.20-35    2017-03-25 CRAN (R 3.3.3)
##  lazyeval       0.2.0      2016-06-12 CRAN (R 3.3.3)
##  magrittr       1.5        2014-11-22 CRAN (R 3.3.3)
##  Matrix         1.2-11     2017-08-16 CRAN (R 3.4.1)
##  MatrixModels   0.4-1      2015-08-22 CRAN (R 3.3.3)
##  memoise        1.1.0      2017-04-21 CRAN (R 3.4.0)
##  methods      * 3.4.2      2017-09-29 local
##  munsell        0.4.3      2016-02-13 CRAN (R 3.3.3)
##  mvtnorm        1.0-6      2017-03-02 CRAN (R 3.3.3)
##  papaja       * 0.1.0.9492 2017-10-27 local
##  parallel       3.4.2      2017-09-29 local
##  pbapply        1.3-3      2017-07-04 CRAN (R 3.4.1)
##  pkgconfig      2.0.1      2017-03-21 CRAN (R 3.4.0)
##  plyr           1.8.4      2016-06-08 CRAN (R 3.3.3)
##  purrr          0.2.3      2017-08-02 CRAN (R 3.4.1)
##  R6             2.2.2      2017-06-17 CRAN (R 3.4.0)
##  RColorBrewer   1.1-2      2014-12-07 CRAN (R 3.3.3)
##  Rcpp           0.12.12    2017-07-15 cran (@0.12.12)
##  readr          1.1.1      2017-05-16 CRAN (R 3.4.1)
##  rgexf          0.15.3     2015-03-24 CRAN (R 3.3.3)
##  rlang          0.1.2      2017-08-09 CRAN (R 3.4.1)
##  rmarkdown      1.6        2017-06-15 CRAN (R 3.4.2)
##  Rook           1.1-1      2014-10-20 CRAN (R 3.3.3)
##  rprojroot      1.2        2017-01-16 CRAN (R 3.4.2)
##  rstudioapi     0.7        2017-09-07 CRAN (R 3.4.1)
##  scales         0.5.0      2017-08-24 CRAN (R 3.4.1)
##  stats        * 3.4.2      2017-09-29 local
##  stringi        1.1.5      2017-04-07 CRAN (R 3.4.0)
##  stringr        1.2.0      2017-02-18 CRAN (R 3.3.3)
##  tibble         1.3.4      2017-08-22 CRAN (R 3.4.1)
##  tidyr          0.7.1      2017-09-01 CRAN (R 3.4.1)
##  tools          3.4.2      2017-09-29 local
##  utils        * 3.4.2      2017-09-29 local
##  viridis        0.4.0      2017-03-27 CRAN (R 3.3.3)
##  viridisLite    0.2.0      2017-03-24 CRAN (R 3.3.3)
##  visNetwork     2.0.1      2017-07-30 CRAN (R 3.4.1)
##  withr          2.0.0      2017-07-28 CRAN (R 3.4.1)
##  XML            3.98-1.9   2017-06-19 CRAN (R 3.4.0)
##  yaml           2.1.14     2016-11-12 CRAN (R 3.4.2)
```

Chapter 1

Introduction

papaja is short for ‘preparing APA journal articles’ and is the name of this R package designed to create fully reproducible journal articles that seamlessly fuse statistical analyses, simulations, and prose. A manuscript written with `papaja` can be thought of as an extensively commented analysis script ready for publication in a scientific journal.

1.1 Motivation

APA style is one of the major style regimes in academia concerned with written scientific communication and is defined in the *Publication Manual of the American Psychological Association* [APA; **?**]. Research in psychology and many fields other, including other social sciences, medicine, and public health is reported in APA style. If you want to publish psychological research, you will have to produce properly formatted APA style manuscripts. While the merits of standardizing scientific reporting are undisputed, mastering the currently 272 pages of Publication Manual’s style definitions puts an undeniable burden on authors. To simplify the writing process researchers have developed strategies to automate the implementation of APA style.

The dominant approach to writing scientific reports in psychology is the use of common word processors such as Microsoft Word or Libre Office. Available APA style document templates set up page margins, line spacing, fonts, and related aspects of the manuscript. The format of citations and references can be automated by using references managers that integrate with the word processor, such as Zotero.

Far less common in psychology, is the use of markup systems, such as LaTeX or Markdown. A key concept of markup systems is to separate style from content by means of document annotations. These annotations declare portions of text for example as title, section headings, or list items but, crucially, they are agnostic to what this means visually (e.g., `<emphasize>text</emphasize>` instead of *text*). There are several advantages to this approach but, crucially for the present purpose, markup systems compile plain text files to produce the final document. Critically, the compilation process is required for fully reproducible scientific reports.

1.1.1 Reproducible scientific reports

Dynamic scientific reports fuse prose and simulation or analysis scripts [**??**]. Each time a dynamic document is compiled, the scripts are executed anew and the results—statistics, tables, and figures—are recreated and inserted into the document. Thereby dynamic documents allow automated reporting of results, which is currently not possible with the prevalent use of Microsoft Word or Libre Office. Importantly, dynamic documents ensure that the results reported in the manuscript are free of rounding or transcription errors and correspond to the performed analyses. This is the reason why dynamic scientific reports are commonly refered to as reproducible scientific reports or reproducible reports. Thus, the term reproducibility here means that given a dataset and analysis script any analyst obtains the same statistical results [**??**].

The importance of reproducibility was pinpointed by the U.S. National Science Foundation subcommittee on replicability in science:

> Reproducibility is a minimum necessary condition for a finding to be believable and informative. [p. 4, **?**]

R Markdown has been suggested as one possible framework for reproducible analyses [**?**]. `papaja` is a R-package in the making including a R Markdown template that can be used with (or without) RStudio to produce documents, which conform to the American Psychological Association (APA) manuscript guidelines (6th Edition). The package uses the LaTeXdocument class apa6 and a .docx-reference file, so you can create PDF documents, or Word documents if you have to. Moreover, `papaja` supplies R-functions that facilitate reporting results of your analyses in accordance with APA guidelines.

Markdown is a simple formatting syntax that can be used to author HTML, PDF, and MS Word documents (among others). In the following I will assume you have hopped onto the band wagon and know how to use R Markdown to conduct and comment your analyses. If this is not the case, I recommend you get to grips with R Markdown first. I use RStudio (which makes use of pandoc) to create my documents, but the general process works using any other text editor.

1.2 Implementation

In this section we provide a quick overview of the software and the underlying design principles. Unless you are interested in these details it's safe to skip this section.

1.2.1 Guiding principles

In our development of this package our design decisions are based on two guiding principles:

1. **`papaja` documents are convertible**. A central motivation for the development of the package was the experience that many (if not most) scientific journals in psychology require that the final version of a submitted manuscript be provided as a Microsoft Word or RTF document. We, thus, strive to employ solutions that work for both LaTeX and Word output formats even if it means that some of the functions are less versatile than they could be.
2. **`papaja` does not compute**. The package is meant to facilitate writing reports and manuscripts in APA style; it is *not* meant to analyse data. Functions provided in this package do as little computing as possible to grant authors maximal analytic flexibility. Moreover, limiting the amount of computing done by the package facilitates package maintenance.

While we are guided by these principles, they are guidelines rather than commandments and we deviate from them if necessary. For example, `papaja` functions do compute statistics that should be routinely reported according to APA guidelines, such as confidence intervals, if they are not provided in the analysis output objects. We, however, try to enable customization and replacement of these defaults whenever we can.

1.2.2 Document compilation

It is important to note that `papaja` builds on several existing R packages, which in turn build on other software, to compile the final document. This layered software design grants the package its capabilities but it comes at a cost: When compilation of a `papaja`-document throws an error it may not be immediately obvious to an inexperienced user, which part of the process failed. We, thus, provide an overview of the compilation process and the software involved.

The compilation process is outlined in Figure 1.1. As detailed in the R Markdown components section, an R Markdown document consists of three components: A YAML front matter that contains document meta data, prose written in markdown (and optionally LaTeX) syntax, and R code in so-called *code chunks*.

Additionally, the R code contained in an R Markdown document usually accesses data that are either stored in a file on the computers hard drive or a database on the local network or internet. When the document is compiled either by calling `rmarkdown::render()` or by clicking the Knit-button in RStudio, the YAML front matter is parsed. Errors during this processing step are usually due to erroneous YAML syntax and start with `Error in yaml::yaml.load(enc2utf8(string), ...) :`. Next, the R code inside the R Markdown document is executed sequentially from top to bottom of the document and the resulting output is inserted into the document in markdown syntax. Any figures that are generated during the execution of the R code are stored to the hard drive and embedded into the markdown document. The R code is executed in a new R session with an empty workspace, no attached packages, and with the working directory set to the path of the R Markdown file. If any portion of the R code throws an error the compilation is aborted. In RStudio the debugger will point to the first line of the R code chunk that threw the error—not the exact line responsible for the error—and report the error message. R error messages usually start with `Error:`, for example, `Error: Object 'experiment_data' not found.`

After the R code has been executed the `bookdown` document format `pdf_document2()` or `word_document2()`, respectively, provides functions that enable the cross-referencing syntax `\@ref()` described in the Cross-referencing section. Incorrect use of the cross-referencing syntax typically does not interfere with the document compilation but is apparent from missing or incorrect cross-references in the compiled document, e.g. instead of a figure number the document may contain the figure handle such as . The result of this processing step is a pure markdown document containing only a YAML header, prose in markdown (and optionally LaTeX) syntax and rendered images in PDF, EPS, PNG, and TIFF formats at 300 dpi as set by `apa6_pdf()` or `apa6_word()`, respectively.

The mardown document, images and an optional BiB(La)TeX file that contains bibliographic information are the input to the next processing step. `rmarkdown::render()` assambles a system call to pandoc that is supplemented by `papaja` to include a call to an additional filter and can be further customized by the user. pandoc parses the contents of the markdown file and translates it into an abstrac syntax tree (AST). The `pandoc-citeproc` filter is than applied to the AST to replace the citation markup by the citation text and insert a reference section. As detailed in the Citations section, `pandoc-citeproc` relies on the citation styles language (CSL), an XML-based specification of citation and bibliography formatting rules. Unfortunately, CSL does not support in-text citations. To overcome this limitation, `pandoc-citeproc` provides a separate syntax for in-text citations, which is based on the same CSL style. As a result the CSL APA6 style uses ampersands for all citations, including in-text citations, defying the APA citation style. That is why `papaja` provides an additional R-based filter that is applied to the AST to replace ampersands in in-text citations with the word *and*.

After all filters have been applied to the AST pandoc converts the document into the target output format, that is, PDF or DOCX. To create DOCX-documents `papaja` provides a reference DOCX-file containing page setup and style definitions that pandoc uses to create the manuscript file. That is, pandoc translates the AST directly to the office open XML format. To create PDF-documents pandoc in turn relies on LaTeX, that is, pandoc translates the AST to a TeX-file based on a templated provided by `papaja`. The TeX-file template invokes the `apa6` LaTeX document class [@???] and loads several additional LaTeX packages. pandoc then generates a system call to LaTeX to render the PDF document. During this processing step, that is, following the pandoc system calls, errors are due to failures of LaTeX to compile the TeX-file and usually start with `!`, for example, `! Undefined control sequence.` or `! Missing $ inserted`.

1.3 Alternatives

Needless to say, not all journals require manuscripts to be prepared according to APA guidelines. Other journal templates for the `rmarkdown` package are available in the `rticles` package. It includes a set of templates for authoring of R related journal and conference submissions.

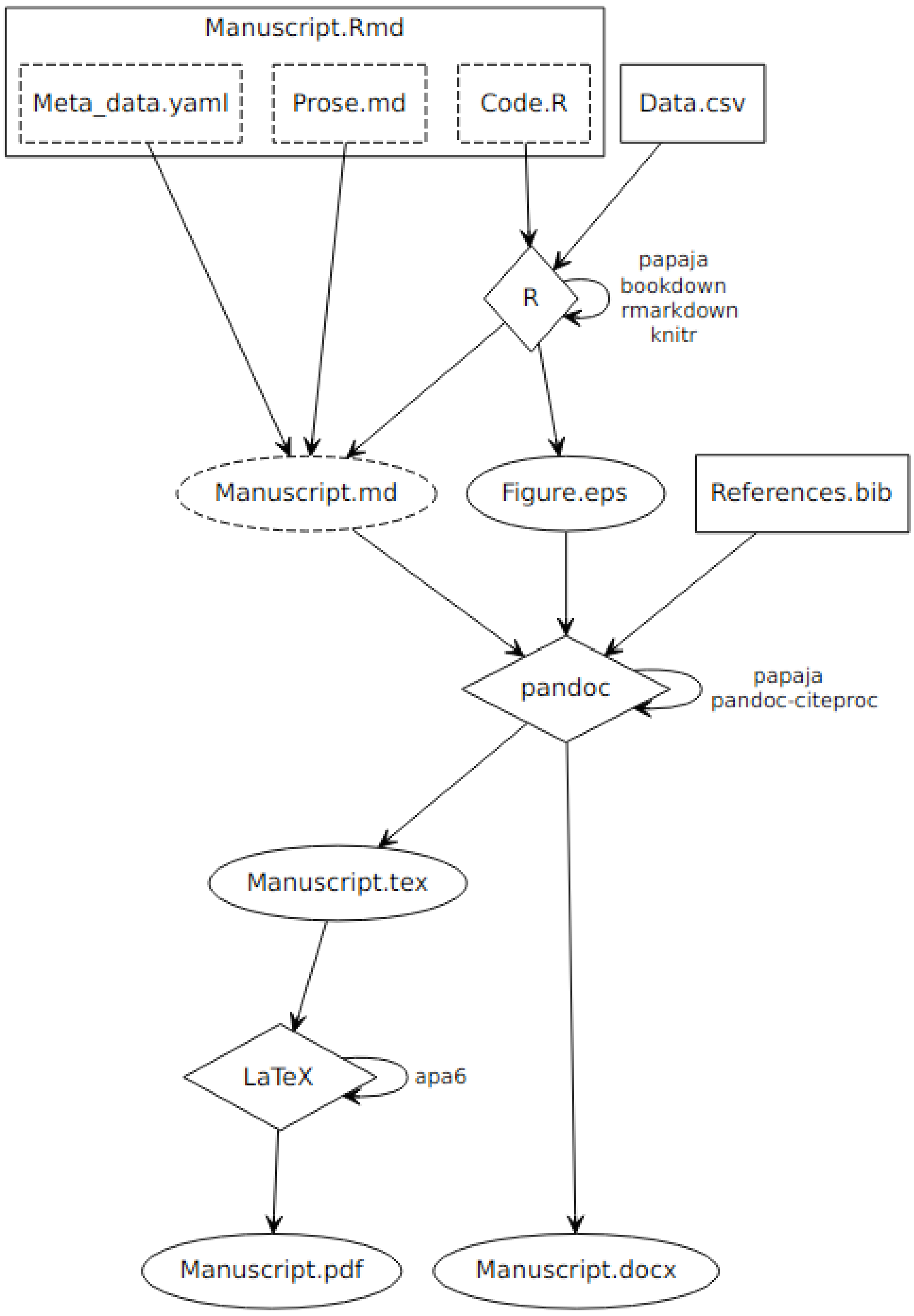

Figure 1.1: Process diagram illustrating the compilation of reproducible reports written with ‘papaja‘. Boxes represent input files, diamonds represent compiling software, and ovals represent output files. Dashed components are either implicit (components of R Markdown files) or by default deleted after the compilation and, thus, invisible to the user.