Prepare APA Journal Articles with R Markdown

Frederik Aust\textsuperscript{1,2} & Marius Barth\textsuperscript{1}

\textsuperscript{1} University of Cologne
\textsuperscript{2} University of Amsterdam

Author Note

The authors made the following contributions. Frederik Aust: Conceptualization, Writing—Original Draft Preparation, Writing—Review & Editing, Software, Project administration; Marius Barth: Conceptualization, Writing—Review & Editing, Software.

Correspondence concerning this article should be addressed to Frederik Aust, Department Psychology, University of Cologne, Herbert-Lewin-Str. 2, 50931 Köln, Germany.

E-mail: frederik.aust@uni-koeln.de
Abstract

papaja addresses computational non-reproducibility in research reports caused by reporting errors, i.e. incomplete or incorrect reporting of the analytic procedure or analytic results. The package is tailored to authors of scientific manuscripts that must adhere to the guidelines of the American Psychological Society (6th edition). This document was written with papaja and provides a brief overview of the package’s main features: An R Markdown template for APA-style manuscripts and helper-functions that facilitate reporting of analytic results in accordance with APA guidelines.

Keywords: APA style, R, knitr, R markdown, papaja

Word count: 3147
Prepare APA Journal Articles with R Markdown

Computational reproducibility is of fundamental importance to the quantitative sciences (Cacioppo, Kaplan, Krosnick, Olds, & Dean, 2015; Donoho, 2010; Hutson, 2018; Peng, 2011). Yet, non-reproducible results are widely prevalent. Computational reproducibility is threatened by countless sources of errors, but among the most common problems are incomplete or incorrect reporting of statistical procedures and results (Artner et al., 2020). papaja was designed to address these problems. The package is tailored to authors of scientific manuscripts that must adhere to the guidelines of the American Psychological Association (APA, 6th edition, American Psychological Association, 2010).

papaja provides rmarkdown (Xie, Allaire, & Grolemund, 2018) templates to create DOCX documents and PDF documents—using \LaTeX document class apa6. Moreover, papaja provides helper functions to facilitate the reporting of results of your analyses in accordance with APA guidelines. This document was written with papaja and provides a brief overview of the package’s main features. For a comprehensive introduction and installation instructions, see the current draft of the papaja manual.\footnote{If you have a specific question that is not answered in the manual, feel free to ask a question on Stack Overflow using the papaja tag. If you believe you have found a bug or would like to request a new feature, open an issue on Github and provide a minimal complete verifiable example.}

The problem: Copy-paste reporting

Readers of scientific journal articles generally assume that numerical results and figures directly flow from the underlying data and analytic procedure. Execution of analyses and reporting of results are typically not considered sources of error that threaten the validity of scientific claims—the computational reproducibility of the reported results is a forgone conclusion. The natural assumption of computational reproducibility reflects its fundamental importance to quantitative sciences as acknowledged 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, Cacioppo et al., 2015)

Non-reproducible results are scientifically and ethically unacceptable. They impede an accumulation of knowledge, waste resources, and when applied could have serious consequences. A recent investigation of breast cancer treatments erroneously concluded that radiotherapy after mastectomy increased mortality because of an error in the analysis code (Henson et al., 2016). A corrected reanalysis indicated that, in fact, the opposite was the case—the treatment appeared to be effectively decrease mortality. Examples like this show that computational reproducibility cannot be a forgone conclusion.

Large-scale scrutiny of statistics published in over 30,000 articles in psychology journals shows that every other article reports at least one impossible combination of test statistic, degrees of freedom, and $p$ value; in every tenth article such inconsistencies call the statistical inference into question (Nuijten, Hartgerink, Assen, Epskamp, & Wicherts, 2016). More in-depth investigations that attempted to reproduce reported results from the underlying raw data paint a similar picture. For example, in a sample of 46 articles, two thirds of key claims could be reproduced but in every tenth case only after deviating from the reported analysis plan (Artner et al., 2020). For one in four non-reproducible results, the reproduction attempt yielded results that were no longer statistically significant, calling the original statistical inference into question. These figures clearly show that there is a need for efforts to improve the computational reproducibility of the published literature.

Computational non-reproducibility is, of course, multi-causal. While there is only one way in which a research report is computationally reproducible, there is a countless number of things that can go wrong. Broadly speaking, there are at least four causes for non-reproducible analyses: (1) incomplete or incorrect reporting of the analytic procedure, (2) incorrect execution of the analytic procedure, (3) incorrect reporting of results, and (4) code rot, i.e., non-reproducible caused by (inadvertent) changes to the computational
environment (e.g., software updates, changes to data files). We currently see no technical
solution to the first two causes. Incomplete reporting (1) may be partially mitigated by
strictly enforcing reporting guidelines. However, verifying that the analytic procedure is
reported faithfully (1) and was executed correctly (2) ultimately requires manual scrutiny of
analysis scripts and/or reproduction and is possible only if authors share their data. Code
rot (4), on the other hand, can be adequately addressed by conserving the software
environment in which the results were produced (e.g., R and all R packages). Several
seasoned technical solutions, such as software containers or a virtual machine, exist (Grüning
et al., 2018; Piccolo & Frampton, 2016).\textsuperscript{2} \textit{papaja} provides a technical safeguard for correct
reporting of results (3).

When it comes to reporting quantitative results, most researchers engage in what we
refer to as \textit{copy-paste reporting}. Quantitative analyses and reporting are done in separate
software. Thus, by necessity quantitative results are copied from the analysis software and
pasted into the report. Copy-paste reporting underlies and contributes to several of the most
common causes for computational non-reproducibility: Rounding errors, incorrect labeling of
statistical results, typos, and inserting results of a different analysis (pp. 12-13, Artner et al.,
2020). We are convinced that errors caused by copy-paste reporting cannot be addressed by
appealing to researchers to be more careful. The motivation to avoid such errors should
already be high because the reputational cost of errata and retractions due to
non-reproducible results is substantial. Even researchers that open their data (and analysis
code) to the public or anticipate systematic editorial scrutiny report non-reproducible results
(Eubank, 2016; Hardwicke et al., 2018; Obels, Lakens, Coles, Gottfried, & Green, 2020).
Evidently, computational reproducibility is difficult to attain.

\textsuperscript{2} \textit{papaja} can be readily combined with these tools as documented in the section on \textit{reproducible software
environments} in the \textit{papaja} manual.
The solution: Dynamic documents

We believe copy-paste reporting is a flawed approach to reporting quantitative results. Hence, we believe researchers need stop copy-pasting to safeguard the computational reproducibility of their manuscripts. Manuscripts should be dynamic (or “living”) documents (Knuth, 1984; Xie et al., 2018) that contain direct links to the analytic software. Dynamic documents fuse analysis code and prose such that statistics, figures, and tables are automatically inserted into a manuscript—and updated as data or analysis code change. As an added benefit, dynamic documents have great potential to improve the computational reproducibility of manuscripts beyond reporting errors as they facilitate independent reproduction. Dynamic documents fully document the analytic procedure and establish direct links to the associated scientific claims.

papaja, and the software it builds on, provides researchers with the tools to create dynamic submission-ready manuscripts in the widely used APA style. The dominant approach to creating dynamic documents in R is to use the rmarkdown package (Xie et al., 2018). papaja provides R Markdown templates to create DOCX and PDF documents (using \LaTeX{}document class apa6). Moreover, papaja provides several functions to conveniently report analytic results according to APA guidelines. The remainder of this document illustrates how these functions can be used.

Setting up a new document

Once papaja and all other required software is installed, the APA template is available through the RStudio menu, see Figure 1. When you click RStudio’s Knit button, a manuscript conforming to APA style is rendered, which includes both your text and the output of any embedded R code chunks within the manuscript. Of course, a new document can also be created without RStudio using rmarkdown::draft() and rendered using rmarkdown::render().
# Create new R Markdown file

```r
rmarkdown::draft(
  "manuscript.Rmd"
  , "apa6"
  , package = "papaja"
  , create_dir = FALSE
  , edit = FALSE
)
```

# Render manuscript

```r
rmarkdown::render("manuscript.Rmd")
```

![New R Markdown dialog box](image)

**Figure 1.** After successful installation the **papaja** APA manuscript template is available via the RStudio menu.

This document is in APA manuscript style, but other styles are available for PDF documents. The document style can be controlled via the `classoption` field of the YAML front matter. For a thesis-like style change `classoption` to `doc` or use `jou` for a more
polished journal-like two-column layout. For a comprehensive overview of other formatting options please refer to the papaja manual.

To create DOCX documents, the output field in the YAML front matter can be set to papaja::apa6_docx. Please note, however, that DOCX documents are somewhat less flexible and less polished than PDF documents. papaja builds on pandoc to render Markdown into PDF and DOCX documents. Unfortunately, pandoc’s capabilities are more limited for DOCX documents. This is why some papaja features are only available for PDF documents, for example, see the summary of rendering options in the manual. Also, DOCX documents require some limited manual work before they fully comply with APA guidelines. The DOCX documents produced by papaja should, however, be suitable for collaboration with colleagues, who prefer Word over R Markdown and to prepare journal submissions.

Writing

Like rmarkdown, papaja uses Markdown syntax to format text. A comprehensive overview of the supported Markdown syntax is available in the pandoc manual. In the following, we will highlight a few features that are of particular relevance to the technical writing of research reports.

Citations

By default, citations in papaja are processed by the pandoc extension citeproc, which works well for both PDF and DOCX documents. citeproc takes reference information from a bibliography file, which can be in one of several formats (e.g., CSL-JSON, Bib(La)TeX, EndNote, RIS, Medline). To start citing, specify the path to the bibliography file in bibliography field of the YAML front matter. Once citeproc knows where to look for reference information, [@james_1890] will render to a citation within parentheses, i.e., (James, 1890). Multiple citations must be separated by a semicolon (; e.g., [@james_1890; @bem_2011]) and are automatically ordered alphabetically as per APA style, i.e., (Bem,
2011; James, 1890). To cite a source in text simply omit the brackets. The pandoc manual provides a comprehensive overview of citeproc and the supported citation syntax.

To facilitate inserting citations, you may use the RStudio Visual Editor’s bibliography search and auto-completion of reference handles. If you use VSCode with the R extension or RStudio without the Visual Editor, the add-in provided in citr serves a similar purpose. Both the Visual Editor and citr can also access your Zotero database directly and copy references to your bibliography file.

As academics and open source developers, we believe it is important to credit the software we use for our publications. A lot of R packages are developed by academics free of charge. As citations are the currency of academia, it is easy to compensate volunteers for their work by citing their R packages. papaja provides two functions that make citing R and its packages quite convenient:

\begin{verbatim}
r_refs() creates a BibLaTeX file containing citations for R and all currently loaded packages. cite_r() takes these citations and turns them into readily reportable text.
\end{verbatim}

\begin{verbatim}
my_citation now contains the following text that you can use in your document:
\end{verbatim}

\begin{verbatim}
R [Version 4.1.2; @R-base] and the R-packages *afex* [Version 1.0.1; @R-afex], *dplyr* [Version 1.0.8; @R-dplyr], *ggforce* [Version 0.3.3; @R-ggforce], *ggplot2* [Version 3.3.5; @R-ggplot2], *lme4* [Version 1.1.28; @R-lme4], *Matrix* [Version 1.3.4; @R-Matrix], *papaja* [Version 0.2.0; @R-papaja], and *tinylabels* [Version 0.2.3; @R-tinylabels]
\end{verbatim}

Equations

Equations can be reported using the powerful \LaTeX syntax. Inline math must be enclosed in $ or \langle and \rangle, for example, $d' = z(H) - z(\text{FA})$, which renders to $d' = z(H) - z(\text{FA})$. For larger formulas, displayed equations are more appropriate; they are
enclosed in $$ or \[and \], and will, for example, render to

\[ d' = \frac{\mu_{old} - \mu_{new}}{\sqrt{0.5(\sigma_{old}^2 + \sigma_{new}^2)}}. \]

**Reporting results**

If you are not familiar with R Markdown and how it can be used to conduct and document your analyses, we recommend you familiarize yourself with R Markdown first. RStudio provides a concise introduction.

`apa_print()` is a core function in `papaja` to facilitate reporting analytic results for a growing number of analytic output objects, Table 1. Consider the following example of an analysis of variance. After performing the analysis, the result is passed to `apa_print()`.

The function takes the R object returned by the analysis function and returns a list that contains reportable text and tables.

```r
recall_anova <- afex::aov_4(
  Recall ~ (Task * Valence * Dosage) + (Task * Valence | Subject),
  data = mixed_data
)
recall_anova_results <- apa_print(recall_anova)
str(recall_anova_results)
```

```
## List of 4
## $ estimate :List of 7
## ..$ Dosage : chr "$\hat{\eta}^2_G = .267$, 90\% CI $[.000, .507]$"
## ..$ Task : chr "$\hat{\eta}^2_G = .048$, 90\% CI $[.000, .297]$"
## ..$ Valence : chr "$\hat{\eta}^2_G = .008$, 90\% CI $[.000, .052]$"
## .. [list output truncated]
## $ statistic :List of 7
## ..$ Dosage : chr "$F(2, 15) = 2.97$, $p = .082$"
```
The text returned by \texttt{apa_print()} can be inserted into manuscript as usual using inline code chunks:

\begin{verbatim}
Item valence (\texttt{\textasciitilde r in_paren(recall_anova_results$full_result$Valence)}) and the task affected recall performance, \texttt{\textasciitilde r recall_anova_results$full_result$Task}; the dosage, however, had no detectable effect on recall, \texttt{\textasciitilde r recall_anova_results$full_result$Dosage}. There was no detectable interaction.
\end{verbatim}

The above excerpt from an R Markdown document yields the following in the rendered document. Note that the function \texttt{in_paren()} replaces parentheses with brackets as per APA guidelines when statistics are reported in parentheses.

Item valence ($F[1.62, 24.36] = 3.46, p = .056, \hat{\eta}^2_G = .008, 90\% \text{CI} [.000, .052]$) and the task affected recall performance, $F(1, 15) = 43.13, p < .001, \hat{\eta}^2_G = .048, 90\% \text{CI} [.000, .297]$; the dosage, however, had no effect on recall, $F(2, 15) = 2.97, p = .082, \hat{\eta}^2_G = .267, 90\% \text{CI} [.000, .507]$. There was no significant interaction.
Table 1

Object classes currently supported by `apa_print()`.

<table>
<thead>
<tr>
<th>A-B</th>
<th>D-L</th>
<th>L-S</th>
<th>S-Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>afex_aov</td>
<td>default</td>
<td>lsmobj</td>
<td>summary.aovlist</td>
</tr>
<tr>
<td>anova</td>
<td>emmGrid</td>
<td>manova</td>
<td>summary.glht*</td>
</tr>
<tr>
<td>anova.lme</td>
<td>glht*</td>
<td>merMod</td>
<td>summary(glm)</td>
</tr>
<tr>
<td>Anova.mlm</td>
<td>glm</td>
<td>mixed</td>
<td>summary.lm</td>
</tr>
<tr>
<td>aov</td>
<td>htest</td>
<td>papaja_wsci</td>
<td>summary.manova</td>
</tr>
<tr>
<td>aovlist</td>
<td>list</td>
<td>summary_emm</td>
<td>summary.ref.grid</td>
</tr>
<tr>
<td>BFBayesFactor*</td>
<td>lm</td>
<td></td>
<td>summary.Anova.mlm</td>
</tr>
<tr>
<td>BFBayesFactorTop*</td>
<td>lme</td>
<td></td>
<td>summary.aov</td>
</tr>
</tbody>
</table>

*Note.* *These methods are not fully tested; don’t trust blindly!*

In addition to individual text strings, `apa_print()` also summarizes all results in a standardized `data.frame`.\(^3\) The column names conform to the naming conventions used in the broom package (e.g. `estimate`, `statistic`, and `p.value`). `apa_print()` assigns each column an additional descriptive variable label.

```r
head(recall_anova_results$table, 3)
```

```r
table
## A data.frame with 7 labelled columns:
##
## term estimate conf.int statistic df df.residual p.value
##
## 1 Dosage .267 [.000, .507] 2.97 2 15 .082
## 2 Task .048 [.000, .297] 43.13 1 15 < .001
## 3 Valence .008 [.000, .052] 3.46 1.62 24.36 .056
```

\(^3\) For more complex analyses the `table` element may contain a named list of multiple tables.
## term : Effect
## estimate : $\hat{\eta}^2_G$
## conf.int : 90\% CI
## statistic: $F$
## df : $\mathit{df}^{\mathrm{GG}}$
## ... (2 more labels)

### Tables

Tables returned by `apa_print()` can be conveniently included in a manuscript by passing them to `apa_table()`. This function was designed with exemplary tables from the APA manual in mind and to work well with `apa_print()`. Conveniently, `apa_table()` uses any available variable labels as informative column headers, Table 2. Unfortunately, table formatting is somewhat limited for DOCX documents due to the limited table representation in `pandoc` (e.g., it is currently not possible span header cells across multiple columns or have multiple header rows). Of course, popular packages for creating tables, such as `kableExtra`, `huxtable`, or `flextable` can also be used and may be preferable for more complex tables.

```r
apa_table(
  recall_anova_results$table,
  caption = "ANOVA table for recall performance as a function of task, valence, and dosage."
  , note = "This is a table created using apa_print() and apa_table()."
  , align = "lrcrlr"
  , midrules = c(3, 6)
)
```

As required by the APA guidelines, tables are deferred to the final pages of the
Table 2

ANOVA table for recall performance as a function of task, valence, and dosage.

<table>
<thead>
<tr>
<th>Effect</th>
<th>$\hat{\eta}^2_G$</th>
<th>90% CI</th>
<th>$F$</th>
<th>$df^G_G$</th>
<th>$df^G_{res}$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosage</td>
<td>.267</td>
<td>[.000, .507]</td>
<td>2.97</td>
<td>2</td>
<td>15</td>
<td>.082</td>
</tr>
<tr>
<td>Task</td>
<td>.048</td>
<td>[.000, .297]</td>
<td>43.13</td>
<td>1</td>
<td>15</td>
<td>&lt; .001</td>
</tr>
<tr>
<td>Valence</td>
<td>.008</td>
<td>[.000, .052]</td>
<td>3.46</td>
<td>1.62</td>
<td>24.36</td>
<td>.056</td>
</tr>
<tr>
<td>Dosage × Task</td>
<td>.004</td>
<td>[.000, .000]</td>
<td>1.83</td>
<td>2</td>
<td>15</td>
<td>.195</td>
</tr>
<tr>
<td>Dosage × Valence</td>
<td>.011</td>
<td>[.000, .000]</td>
<td>2.38</td>
<td>3.25</td>
<td>24.36</td>
<td>.090</td>
</tr>
<tr>
<td>Task × Valence</td>
<td>.003</td>
<td>[.000, .000]</td>
<td>1.50</td>
<td>1.35</td>
<td>20.20</td>
<td>.242</td>
</tr>
<tr>
<td>Dosage × Task × Valence</td>
<td>.001</td>
<td>[.000, .000]</td>
<td>0.39</td>
<td>2.69</td>
<td>20.20</td>
<td>.743</td>
</tr>
</tbody>
</table>

Note. This is a table created using `apa_print()` and `apa_table()`.

manuscript when creating PDF documents. To place tables and figures in text instead, the `floatsintext` field in the YAML header can be set to `yes`.

Figures

Figures generated in R are automatically inserted into the document. `papaja` provides a set of functions built around `apa_factorial_plot()` that facilitate visualizing data from factorial study designs, Figure 2(A). For `ggplot2` users, `papaja` provides `theme_apa()`, a theme designed with APA manuscript guidelines in mind, Figure 2(B).

```r
apa_beeplot(
  mixed_data,
  id = "Subject",
  dv = "Recall",
  factors = c("Valence", "Dosage", "Task")
)```

Again, this is currently not the case in DOCX documents.
Again, as required by the APA guidelines, figures are deferred to the final pages of the document unless the `floatsintext` field in the YAML header can be set to `yes`.

Referencing tables and figures

`papaja` builds on the `bookdown` package, which provides limited cross-referencing capabilities within documents. By default, automatically generated table and figure numbers can be inserted into the text using `@ref(tab:chunk-name)` for tables or `@ref(fig:chunk-name)` for figures. Note that for this syntax to work chunk names cannot include underscores (i.e., `_`).

Getting help

For a comprehensive introduction to `papaja`, check out the current draft of the `papaja` manual. If you have a specific question that is not answered in the manual, feel free to ask a question on Stack Overflow using the `papaja` tag. If you believe you have found a bug or you want to request a new feature, open an issue on Github and provide a minimal complete verifiable example.

If you are interested to see how others use `papaja`, take a look at some of the publicly available R Markdown files. The file used to create this document is available at the `papaja` GitHub repository. Moreover, a collection of papers written with `papaja`, including the corresponding R Markdown files, is listed in the manual. If you have published a paper that
Figure 2. Bee plots of the example data set. Small points represent individual observations, large points represent means, and error bars represent 95% confidence intervals.
was written with **papaja**, please add the reference to the public Zotero group yourself or send us to me.

**Contributing**

If you like **papaja** and would like to contribute, we highly appreciate any contributions to the R package or its documentation. Take a look at the open issues if you need inspiration. There are many additional analyses that we would like `apa_print()` to support; new S3/S4-methods are always appreciated (e.g., for `factanal`, `fa`, `lavaan`). For a primer on adding new `apa_print()`-methods, see the getting-started-vignette (`vignette("extending_apa_print", package = "papaja")`). Before working on a contribution, please review our brief contributing guidelines and code of conduct.

Enjoy writing. :)}
References


