Rmarkdown for Beginners

Part 1: Word Documents

Mertcan Güngör

Dec 27, 2019

This is a beginner’s guide to making dynamic and reproducible .docx files in RMarkdown. It will teach you about the packages that allow us to easily insert APA-formatted results and tables into your text. It may not be enough to produce a fully-formatted APA style manuscript by itself, due to limited functioning of some packages in Word documents. And as a Microsoft Word or Google Docs user, you may not want to write your whole paper in RStudio anyway. What really matters is making your results **reproducible** and **error-proof**. So if you only want to use RMarkdown for your Results section and do the rest of the paper in another software, that’s fine! However, if you want your whole document to be fully-reproducible and maybe try the polished journal article look, I suggest you check out the second guide on *papaja* pdf templates after you finish this one.

## R + Markdown

RMarkdown is a useful format that integrates code into text. As the subheading suggests, the code follows R rules, whereas the text follows Markdown rules. Markdown is a markup language (like HTML) which uses a simple syntax to turn the plain text in your .Rmd file into rich text in your output file (in this case, .docx). For example, using asterisks and underscores, you can have your text in *italic*, **bold**, or in ***both***. You can use squared brackets to put [hyperlinks](https://osf.io/w39vn/) and footnotes.[[1]](#footnote-22) You can have different levels of headings using the hashtag sign:

# This is a Level 1 heading

## This one is Level 2

### And this is Level 3

You can check out the “RMarkdown Cheat Sheet” and “RMarkdown Reference Guide” for more syntax tips (In RStudio, go to Help > Cheatsheets). But what about formatting other features, such as the font size, color, alignment, spacing, etc.? For this, you can create another .docx file that will serve as a style template for the .Rmd file. Here’s how:

1. Create a new .Rmd file in Word output format
2. Knit to Word
3. Open the .docx file in Word
4. Open the “Styles” menu and edit each unit (title, main text, Heading 1, Heading 2, etc.) as you want
5. Save the .docx file under a new title (e.g. styletemplate.docx)
6. Open the .Rmd file where you want to do your real work
7. Change the output: word\_document line at the beginning to this:

output:

Word\_document:

reference\_docx: styletemplate.docx

Here’s a short video on how to do it: <https://vimeo.com/110804387> If you want your report in APA style, you can check out the template made by Andrew Jessop: <https://github.com/andrew-jessop/apa-word-docx-in-rmarkdown/blob/master/apa_style.docx>

## Chunks

In RMarkdown, R codes are kept in “chunks”, such as the one below. You can insert a chunk by clicking Insert > R or Code > Insert Chunk or Ctrl + Alt + I. The first chunk is a good place to load all the packages you’ll use in the document. This way, the person who wants to reproduce your document can see which packages are required. Keep in mind that R will run all of the chunks every time you knit the .Rmd, so having install.packages() there will be disastrous. If some packages you use aren’t easily found, you may put the necessary instructions in the comments (see below).

# Don't forget to set the default encoding to UTF-8 (in RStudio, Tools > Global Options > Code > Saving, you may need to restart RStudio afterwards)  
# Don't forget to install the packages you don't have before knitting!  
# papaja is not on CRAN yet, so you need to use devtools::install\_github("crsh/papaja") - please run the code in your console, not in the document.  
# To knit your document in .pdf, you'll need LaTeX. If you haven't installed any LaTeX distribution (like MiKTeX), installing the "tinytex" package will be sufficient. Remember that after you install it from CRAN, you also need to run tinytex::install\_tinytex() in your console to complete the installation.  
library("papaja")  
library("MOTE")  
library("knitr")  
library("psych")  
library("apaTables")  
knitr::opts\_chunk$set(  
 echo = TRUE,  
 message = FALSE,  
 warning = FALSE,  
 results = "asis")

The knitr::opts\_chunk$set() code in this first chunk sets the default options for all the other chunks. These options can be modified for the following individual chunks by typing the necessary commands in the curly brackets on top of each chunk, or by clicking the gear symbol on the upper-right side of the chunk. For example, you can tell *knitr* to show/hide your code using the ìnclude and echo options. I used echo = TRUE here so you can see what each function does, but you’ll probably want to hide the code with echo = FALSE most of the time. You can use eval = FALSE for the chunks you don’t want to run but still want to show in your document (e.g. time-consuming analyses or simulations). You can suppress the warnings and messages other than the output using the messages = FALSE and warnings = FALSE option. results = 'asis' is a must if you want your tables to render correctly.

## Reporting your results

Now we’ll produce some output. Let’s say we want to report the mean and standard deviation of department ratings from the Chatterjee-Price “attitude” data.

descr <- describe(attitude)  
  
descr$mean[1]

[1] 64.63333

descr$sd[1]

[1] 12.17256

To integrate this output into our text, we need to write the object or function within backticks, like this: `r descr$mean[1]`. We’d also want to limit the number of decimals or maybe drop the zeroes before the decimal point (e.g. when reporting correlation coefficients or p-values); printnum() and apa() functions (from *papaja* and *MOTE* packages, respectively) come in handy for this. We only need to write the object inside the function, like the example below:

**Example**: = 64.63, = 12.17

**TIP**: Notice that we used the equation syntax (dollar signs) instead of italics, to make the notation fancier. We can also add Greek letters (e.g. , , , , ) using the LaTeX-based mathematical typesetting.

Now we’ll see how to report the results of our statistical analyses. *papaja*’s apa\_print() function allows us to report correlations, t-tests, ANOVAs and regressions in APA format. Let’s start with a correlation from the “attitude” data:

corr1 <- cor.test(attitude$rating, attitude$complaints, method = "pearson", conf.level = .95)  
  
corr1a <- apa\_print(corr1)  
  
corr1a

$estimate [1] "$r = .83$, 95\% CI , "

$statistic [1] "$t(28) = 7.74$, "

$full\_result [1] "$r = .83$, 95\% CI , , , "

$table NULL

attr(,“class”) [1] “apa\_results” “list”

As you can see, apa\_print() offers some alternatives. Pick the one you find the most informative! ($table doesn’t return anything for correlations and t-tests, but we’ll use it later for ANOVAs and regressions)

**Example**: Handling of employee complaints strongly correlated with the overall rating of the department, , 95% CI , , , .

Let’s do a between-subjects t-test using the built-in “sleep” data:

sleept <- t.test(sleep$extra ~ sleep$group)  
  
sleepta <- apa\_print(sleept)  
  
sleepta

$estimate [1] "$\Delta M = -1.58$, 95\% CI , "

$statistic [1] "$t(17.78) = -1.86$, "

$full\_result [1] "$\Delta M = -1.58$, 95\% CI , , , "

$table NULL

attr(,“class”) [1] “apa\_results” “list”

**Example**: “We could not detect a difference between the two treatment groups, , 95% CI , , , .”

Let’s go back to the “attitude” data and do a multiple regression:

regres <- lm(rating ~ complaints + privileges + learning + raises, data = attitude)  
  
regresa <- apa\_print(regres)

Here, I did not print the whole apa\_print() output because it is much longer. If you run it in your console, you’ll see that it gives us the same options ($estimate, $statistic and $full\_result) for each predictor as well as the overall model.

**Example**: Multiple regression analysis showed that while handling of complaints positively predicted overall department rating, , 95% CI , , , , there was no additional variance explained by not allowing special privileges, , , opportunity to learn, , , or raises, , .

Now we’ll conduct a factorial ANOVA using the “ToothGrowth” dataset. apa\_print() output is also very lengthy here, so let’s not print it.

toothaov <- aov(formula = len ~ supp + dose + supp:dose, data = ToothGrowth)  
  
toothaova <- apa\_print(toothaov)

**Example**: A 3 x 2 ANOVA revealed main effects of dosage, , , , , and delivery method, , , , , of Vitamin C on tooth growth in guinea pigs. These main effects were qualified by an interaction effect, , , , .

## Tables

The results you get may look tidy on your console, but they will turn into a mess once you knit your file. The best way is to put them through some table-making function, such as apa\_table() from *papaja*, kable() from *knitr*, or apa.aov.table(), apa.reg.table() and apa.cor.table() from *apaTables*. You will need to do some tweaking to get the best results; some of these functions require you to do this within the .Rmd, whereas others require you to edit the knitted Word output. If you want your final document to be fully reproducible from your code alone, go for the former option. If you don’t mind editing your Word file as long as the numbers in it are reproducible and error-free, choose the latter.

apa\_table() and kable() can turn matrices or data frames into tables. This means that we can make custom tables from scratch, but let’s use the data frames provided by apa\_print().

kable(toothaova$table, caption = "Dependent Variable: Tooth length")

Dependent Variable: Tooth length

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Effect | F | df1 | df2 | MSE | p | ges |
| Supp | 12.32 | 1 | 56 | 16.67 | .001 | .180 |
| Dose | 133.42 | 1 | 56 | 16.67 | < .001 | .704 |
| Supp Dose | 5.33 | 1 | 56 | 16.67 | .025 | .087 |

apa\_table(toothaova$table, caption = "Dependent Variable: Tooth length", note = "The results are based on the 'ToothGrowth' data from Crampton, E. W. (1947)")

(#tab:anovatable)

*Dependent Variable: Tooth length*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Effect |  |  |  |  |  |  |
| Supp | 12.32 | 1 | 56 | 16.67 | .001 | .180 |
| Dose | 133.42 | 1 | 56 | 16.67 | < .001 | .704 |
| Supp Dose | 5.33 | 1 | 56 | 16.67 | .025 | .087 |

*Note.* The results are based on the ‘ToothGrowth’ data from Crampton, E. W. (1947)

As you can see, both tables are quite similar, since apa\_table() is built on kable(). Assuming we want the tables in APA style, apa\_table() seems a bit better at first. If you look at the column names, you’ll notice that apa\_table() has formatted the apa\_print() output in APA style, whereas kable() has left them raw. apa\_table() also allows you to put a note at the bottom. However, apa\_table() has some problems on Word documents that it doesn’t have on .pdf. One of them is that the horizontal rules which are supposed to be on top and bottom ends of the table are missing. Also, it prints the (#tab:anovatable) line before the table, which we need to remove after we knit to Word.

The choice comes down to the preferences that I mentioned. If you don’t want your document to require edits after knitting, you can go with kable() and customize it to your liking. Check out [this](https://rmarkdown.rstudio.com/articles_docx.html) guide by Richard Layton on how to edit the table style using a .docx template. If editing the knitted file is okay for you, apa\_table() might be a better choice, as it’s integrated with apa\_print() output. Before using either of them, check out their R documentation pages to see the options for editing alignment, font size, etc.

If you have no problem editing the final document, and if you aren’t making a custom table, I think *apaTables* is a better choice given the limitations of apa\_table() in Word documents. It creates properly-formatted tables in separate .doc (or .rtf) files, which needs no extra work other than copying and pasting it to your working document (you can put the table directly into text with filename = NA but it will look messy). I put the code to make an ANOVA table below. The chunk doesn’t run the code (notice the eval=FALSE option) so that it doesn’t produce a file everytime the .Rmd is knitted; run the code yourself and see how nice it is.

apa.aov.table(toothaov, filename = "tooth.doc", table.number = 1)

Let’s try the regression table that apa\_print() gave us:

kable(regresa$table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| predictor | estimate | ci | statistic | p.value |
| Intercept | 11.83 | , | 1.39 | .178 |
| Complaints | 0.69 | , | 4.75 | < .001 |
| Privileges | -0.10 | , | -0.78 | .443 |
| Learning | 0.25 | , | 1.60 | .123 |
| Raises | -0.03 | , | -0.14 | .891 |

apa\_table(regresa$table)

(#tab:table)

\*\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predictor |  | 95% CI |  |  |
| Intercept | 11.83 | , | 1.39 | .178 |
| Complaints | 0.69 | , | 4.75 | < .001 |
| Privileges | -0.10 | , | -0.78 | .443 |
| Learning | 0.25 | , | 1.60 | .123 |
| Raises | -0.03 | , | -0.14 | .891 |

These look similar to the ones we got for the ANOVA. If you want to make a table for hierarchical regression, you’ll need to merge the $table output of multiple models, and then put the final data frame through apa\_table() and kable(). Once again, I will recommend *apaTables*; you only need to put the lm() objects (regression blocks) in the apa.reg.table() function. Run the code below and see for yourself:

regres0 <- lm(rating ~ complaints, data = attitude)  
  
regro <- apa.reg.table(regres0, regres, filename = "regro.doc", table.number = 2)

If you want to use apa\_table() or kable() for a correlation matrix, you’ll need to create and format the matrix using another function such as cor() (you can find a tutorial [here](http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software)). But again, *apaTables* provides a much better option if you don’t mind copy-pasting. Just put the data frame into apa.cor.table() and you’ll get an APA-style correlation matrix which includes the mean and standard deviation for each variable.

apa.cor.table(attitude, filename = "matrix.doc", table.number = 3)

## Bonus: Plots and References

This guide didn’t touch on plots so far, because I don’t have enough experience to teach them and you won’t have much of a problem integrating them into your text anyway. Making the best plot will be more of an R challenge than an RMarkdown challenge. APA rules are also not that strict for figures. However, I just wanted you to remind you that *papaja* has a theme\_apa() function that you can use with *ggplot2* to create APA-styled plots. If you’re not familiar with *ggplot2*, you can check out some basic intros [here](https://sakaluk.wordpress.com/2015/04/13/2-the-ggplot2-package-your-gateway-drug-to-becoming-an-r-user/) and [here](http://egret.psychol.cam.ac.uk/statistics/R/graphs2.html). You can also use interactive apps that generate the R code for you, such as [this one](https://huygens.science.uva.nl/ggPlotteR/).

If you want to write more than just the Results section in RMarkdown, you’ll probably want to cite others. For this, you first need to create a .bib file that contains your references. Reference managers can easily export the library you choose into a .bib. Then, you need to add the bibliography: ["myreferences.bib"] line to the **YAML front matter** that comes before the main text in your .Rmd (the part where you see title, author, etc.). This will put a reference list at the end of your document. To have your references in APA style, you’ll need a .csl (Citation Style Language) file. Download one [here](https://www.zotero.org/styles?q=apa) and just add the line csl: apacslfile.csl below bibliography: in the YAML front matter.

.bib file will give each reference a tag like @Gungor\_2019 that you can use for in-text citations. In the knitted document, [@Gungor\_2019] will look like (Gungor, 2019), @Gungor\_2019 will look like Gungor (2019), and [-@Gungor\_2019] will only give the year in parentheses (2019). Thanks to [citr package](https://github.com/crsh/citr), you can add references into your text with a few clicks, much like using the Word or Docs add-ons for reference managers.

**OPTIONAL (but highly recommended)**: A really cool feature of RMarkdown is that you can easily cite the R packages you use. In the chunk below, you can see the code for generating R references. I’ve already put the filename in the front matter of this .Rmd,[[2]](#footnote-39) so we’ll see the packages at the end. But also within the text body, you can declare how much you’re thankful for R (Version 3.6.3; R Core Team 2020).

r\_refs(file = "rwordref.bib")

## Other helpful resources

* Comprehensive [RMarkdown](https://bookdown.org/yihui/rmarkdown/) guide by Yihui Xie, J. J. Allaire, Garrett Grolemund
* [knitr](https://yihui.org/knitr/) guide by Yihui Xie
* [papaja](https://crsh.github.io/papaja_man/index.html) guide by Frederik Aust & Marius Barth

## References

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

1. Like this! [↑](#footnote-ref-22)
2. RMarkdown can merge the references from multiple .bib files together. You’ll just need to write bibliography: ["myreferences.bib", "rwordref.bib"] [↑](#footnote-ref-39)