Rmarkdown for Beginners - Part 2: Using the *papaja* template

Mertcan Gungor

Bo<U+011F>azici University

Author Note

Mertcan Gungor, Psychological Sciences Graduate Program, Bo<U+011F>azici University, Istanbul, Turkey.

The files related to this guide and the first one can be found at <https://osf.io/w39vn/>.

Correspondence concerning this article should be addressed to Mertcan Gungor, Department of Psychology, Bo<U+011F>azici University, 34342, Bebek/<U+0130>stanbul, Turkey. E-mail: mertcan.gungor@boun.edu.tr

Abstract

This is the second part of the beginner's guide to RMarkdown. The first part was about the basics of the Markdown syntax, embedding your results into text, and some useful packages that give us APA-styled tables and results in Word documents. If you haven't checked out the first one, please do so before you read this one. If you already have, just skip ahead to the main text!

Keywords: Rmarkdown, papaja, reproducibility

Word count: 42069

Rmarkdown for Beginners - Part 2: Using the *papaja* template

Welcome to the second installment of RMarkdown for Beginners. This guide assumes that you've already read the first guide, which goes over the basics of making dynamic Word documents. The basics (how chunks work, how the results are presented, etc.) still apply when we're working with .pdf files; we're going to use the same packages as well. In this guide, I'll talk about the particular benefits and challenges of using *papaja*'s .pdf template for APA-style manuscripts. Let's remember the pros and cons of working with .docx first, so we can compare it to what the *papaja* template offers.

Pros of working with .docx:

- Most people are familiar with Word/Docs
  - Collaboration is easy
  - Not intimidating for new users
- Whatever you don't like in the knitted document (due to limitations of some packages, etc.) can be easily fixed later

Cons of working with .docx:

- To make an APA-style document in Word, you need to use a style template
- To get the best results, you'll probably *need* to edit the knitted document
  - Extra steps between your .Rmd and the final version

Let's look at the pros of using the *papaja* template:

- It takes care of the styling for you
- `apa_table()` function works as intended, so you can have nice-looking tables readily embedded in the document (remember that our best option for Word, *apaTables*, gives us separate documents)

- The document can be fully-reproduced from the .Rmd without extra work

Cons of using the *papaja* template:

- You'll need to do more work within chunks (e.g. tweaking `apa_table()` options)
- You may need to use LaTeX commands before or after you knit the document to get the best results

This guide will show you the options that give you the pros, as well as some ways of working around the cons. But ultimately, the choice is up to you!

Document options

Let's start with finding this particular template, because it's not really out in the open. You first need to click `File > New File > RMarkdown... > From Template` and then choose the `APA article (6th edition) {papaja}` from the list. You'll notice that it already has the section headings for an APA manuscript (Methods, Results, Discussion, etc.); I removed them for this guide. You'll also see that the YAML front matter fields about the title page and the abstract are quite straightforward. You've already learned how to use the `bibliography:` in the first guide; you won't need the `csl:` line here since the references are already in APA style.

The other options need a bit more explanation, so let's quickly go over them. `floatsintext: yes` puts the figures and tables within your text, as opposed to listing them at the end. Although they don't appear directly after the chunk like in Word documents, they are somewhat close. `figurelist: yes` and `tablelist: yes` give you a list of figure and table captions after the reference list. The actual figures and tables come after these lists if you choose the `floatsintext: no` option. `linenumbers: yes` gives you line numbers on the left margin. `mask: yes` removes the author names, affiliations and notes, so that the manuscript is anonymized for peer review. `draft: yes` puts a

watermark that says "DRAFT". `documentclass: "apa6"` is what we're here for, so don't touch that. `classoption: "man"` gives you a double-spaced APA-style manuscript. `classoption: "doc"` will give you a (IMHO) nicer-looking, single-spaced document. If you don't have big tables or plots, definitely check out `classoption: "jou"`: It gives you a two-column document that looks really similar to a journal article. If you want to add an appendix (as an .Rmd file), you can add the line `appendix: myappendix.rmd`. You can check out the other `apa6` options here: http://mirror.las.iastate.edu/tex-archive/macros/latex/contrib/apa6/apa6.pdf.

Tables using `apa_table()`

As I've said before, the use of chunks and reporting results with `apa_print()` work here the same way they do in Word documents; there's no need to repeat them. The key difference is that `apa_table()` function works properly in .pdf files, so we can depend on it. Editing the document after knitting is not ideal here, so we have to do the necessary work within the .Rmd. This includes setting the correct options within the function and manipulating the data frames we use.

`apa_table()` can turn any matrix or data frame into an APA-style table. Let's start with the ANOVA table that `apa_print()` gives us.

```
toothaov <- aov(formula = len ~ supp + dose + supp:dose, data = ToothGrowth)


toothaova <- apa_print(toothaov)


apa_table(toothaova$table, caption = "Dependent Variable: Tooth length",
  note = "The results are based on the 'ToothGrowth' data from Crampton, E. W. (1947)",
  align = c("l", "r", "r", "r", "r", "r", "r"),
  # "placement= h" ensures that tables appear at the exact location
```

```
  # (right after the code, but it can skip to the next page if there's no room).
  # You can also put them at the top, bottom, or on a new page using "placement=".
  # If "floatsintext: no" you don't really have to worry about it.
  placement= "h")
```

Table 1

Dependent Variable: Tooth length

Effect	F	df_1	df_2	MSE	p	$\hat{\eta}^2_G$
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)

Visually, Table 1 is quite nice. `apa_table()` automatically formats the column names (for `apa_print()` tables) and gives it a table number. The only "extra" work needed was to right-align the numbered columns using the `align =` argument. But even though you like the looks, this may not be the information you want to present. Those who ran the *apaTables* functions in the first guide will remember that the `apa.aov.table()` function does things differently than `apa_print()`. This includes displaying SS and MS instead of MSE, using partial eta-square instead of generalized eta-squared, and also giving confidence intervals for the effect size. If you prefer showing these, you can put the `apa.aov.table()` table contents through `apa_table()`. We'll need to modify it a bit, but let's first look at what it gives us at Table 2.

```
tooth <- apa.aov.table(toothaov, conf.level = .95)


apa_table(tooth$table_body, caption = "Dependent Variable: Tooth length",
  note = "The results are based on the 'ToothGrowth' data from Crampton, E. W. (1947)",
  align = c("l", "r", "r", "r", "r", "r", "r", "r"),
  placement= "h")
```

Table 2

Dependent Variable: Tooth length

Predictor	SS	df	MS	F	p	partial_eta2	CI_95_partial_eta2
(Intercept)	889.35	1	889.35	53.34	.000		
supp	227.15	1	227.15	13.62	.001	.20	[.04, .36]
dose	711.88	1	711.88	42.70	.000	.43	[.23, .57]
supp x dose	88.92	1	88.92	5.33	.025	.09	[.00, .24]
Error	933.63	56	16.67				

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

As you can see, the content in Table 2 needs some work. P-values need to be formatted properly but we only have the badly-formatted string versions in the `apa.aov.table()` output, not numeric versions. Remember that `apa_print()` already gave us the formatted versions, so we'll steal them instead.[1] Intercept may not be so

[1] Of course the best strategy is to find the raw p-values and format them yourself. But you should know that different functions use different types of sums of squares, so the numbers may differ for the same ANOVA model. For instance, `aov()` uses Type-1 SS, whereas `lm()` uses Type-3 SS. I used `aov()` just for demonstration but you might want to check out alternative ANOVA functions that use Type-3 SS and are compatible with `apa_print()`:
https://crsh.github.io/papaja_man/reporting.html#anova-anova-anova.mlm-analysis-of-variance-anova

informative for ANOVA, so let's omit it altogether. You'll probably want your variable names capitalized as well. These will be quite easy.

```
toothtable <- tooth$table_body


# Delete the "(Intercept)" line:
toothtable <- toothtable[-1, ]


# Swap the p-value for "dose" with the formatted version from our apa_print() object
toothtable[2,6] <- as.character(toothaova$table$p[2])


# Rename the rows
toothtable[ ,1] <- c("Supp", "Dose", "Supp * Dose", "Error")
```

Now that these are out of the way, the only thing left is to format the column names. This will also be easy to do but let me first explain how. When I talked about pros and cons above, I mentioned using LaTeX commands. LaTeX is a markup language like Markdown. If you've run `apa_table()` in a chunk or console, you have probably noticed the syntax. When we knit to .pdf, *knitr* turns our text into LaTeX (generating a .tex file) and then creates the .pdf based on that. This means that we can use LaTeX syntax to style our document in ways that we can't do with RMarkdown only. In fact, when you open a new *papaja* template and scroll to the bottom, you'll see LaTeX code for formatting the "References" section. But now, instead of putting LaTeX syntax in the text area, we will put it inside the chunk so we can format the output.

To format the column names, we'll use the mathematical typesetting that we used in the first guide. As you may remember, we need dollar signs and backslashes; these are "special characters" that mean something within the syntax. But it can get a bit tricky when the same character (e.g., backslash) is used in both R and LateX syntax. Sometimes

you'll need them to be a part of the syntax, sometimes you'll want them to "escape" (i. e. appear as they are). If you don't properly escape them, they will be taken as badly-written code and you'll get an error message.

Let's start with the knitted document and work backwards. We want our table to have formatted text in the final .pdf document. So we want our preferred formatting to be included as LaTeX syntax in the .tex document that will produce the .pdf. If we want η in the knitted document, we need `$\eta$` to appear in the .tex file. LaTeX code for the table will be generated by the `apa_table()` function, so that's where we want to smuggle our code through. However, `apa_table()` assumes that you want to write normal text by default. Whenever you put special characters in the table content, it will put the necessary code to escape the character (e.g., a backslash will turn to `\textbackslash` in the .tex output). To prevent this, you'll need to use the `escape = F` argument (TRUE by default). But even before we pull out our LaTeX syntax from the chunk into the .tex, we must be able to write it inside the chunk. R won't let us have the single backslash in a string, so we'll have to put two of them to escape a single backslash because that's what we want to put in `apa_table()`. We want the backslash escaped in the R chunk, but we don't want it escaped in the LaTeX output. Check out what I did in the next chunk and see the Table 3 for results.

```
colnames(toothtable) <- c("Predictor", "$SS$", "$df$", "$MS$", "$F$", "$p$",
                          "$\\eta^{2}_{partial}$", "$\\eta^{2}_{partial}$ 95\\% CI")


### Notice that we had to put two backslashes before the percent sign as well.
### That's because % is also a special character for LaTeX! But this time
### we want it to escape. Since we'll choose the "escape = F", we have to
### escape it from LaTeX manually using a backslash, which we also need
### to escape from R!
```

```
### Here's how it goes:
### \\% in R code ---> \% in LaTeX code ---> % in knitted text

apa_table(toothtable, caption = "Dependent Variable: Tooth length",
  note = "The results are based on the 'ToothGrowth' data from Crampton, E. W. (1947)",
  align = c("l", "r", "r", "r", "r", "r", "r", "c"), escape = F, row.names = F,
  # after manipulating the data frame, apa_table() started to show
  # the row numbers, so we had to choose "rownames = F"
  placement= "h")
```

Table 3

Dependent Variable: Tooth length

Predictor	SS	df	MS	F	p	$\eta^2_{partial}$	$\eta^2_{partial}$ 95% CI
Supp	227.15	1	227.15	13.62	.001	.20	[.04, .36]
Dose	711.88	1	711.88	42.70	$<$ .001	.43	[.23, .57]
Supp * Dose	88.92	1	88.92	5.33	.025	.09	[.00, .24]
Error	933.63	56	16.67				

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

TIP: Although we only needed the equation syntax for Table 3, keep in mind that you can use LaTeX commands for other kinds of typesetting (e.g. bold, italics, underline, etc.) in yor tables as well. Make sure you choose `escape = F` and put the right characters to escape it from R. For instance, you should type `\\textbf{bold text}` in the R chunk,

so it becomes `\textbf{bold text}` in LaTeX code and you get **bold text** in the knitted document. Here's a simple guide for the basic commands: http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf

---

Let's now take the regression table from `apa_print()` and put it through `apa_table()`. See Table 4.

```
regres <- lm(rating ~ complaints + privileges + learning + raises, data = attitude)


regresa <- apa_print(regres)


apa_table(regresa$table, caption= "Dependent Variable: Overall department ratings",
          note= "The results are based on Chatterjee-Price Attitude Data.",
          align= c("l", "r", "c", "r", "r"),
          placement = "h")
```

Just like the ANOVA table, Table 4 looks nice and tidy. But depending on the information you want to present, you might find this somehow limited. You may want to give standardized regression coefficients, for example. And as I said in the first guide, you'll need to merge multiple tables if you want to report a hierarchical regression from `apa_print()` output. So let's just take a look at what `apa.reg.table()` offers.

```
regres0 <- lm(rating ~ complaints, data = attitude)


regro <- apa.reg.table(regres0, regres, filename = NA)


apa_table(regro$table_body, font_size = "footnotesize",
          caption= "Dependent Variable: Overall department ratings",
```

Table 4

Dependent Variable: Overall department ratings

Predictor	*b*	95% CI	*t*(25)	*p*
Intercept	11.83	[−5.74, 29.41]	1.39	.178
Complaints	0.69	[0.39, 0.99]	4.75	< .001
Privileges	-0.10	[−0.37, 0.17]	-0.78	.443
Learning	0.25	[−0.07, 0.56]	1.60	.123
Raises	-0.03	[−0.40, 0.35]	-0.14	.891

Note. The results are based on Chatterjee-Price Attitude Data.

```
          note= regro$table_note,
          align= c("l", "r", "c", "r", "c", "r", "c", "l", "l"),
          placement = "p")
```

Without editing, Table 5 looks pretty much unusable here. In addition to poorly formatted column names, it doesn't fit the page even when we use `"footnotesize"` font; this is because it's supposed to be in landscape format. But don't worry, we can always prune the parts we don't need. It's probably a good idea to omit the zero-order correlations, squared semi-partial correlations and their confidence intervals, since these can be reported elsewhere. While we're at it, let's format the column names as well, now that we know how to do it easily.

```
regrot <- regro$table_body


# Remove the sr2, sr2 CI, and r columns
regrot <- regrot[ , -6:-8]
```

Table 5

Dependent Variable: Overall department ratings

Predictor	b	b_95%_CI	beta	beta_95%_CI	sr2	sr2_95%_CI	r	Fit	Differenc
(Intercept)	14.38*	[0.82, 27.94]							
complaints	0.75**	[0.55, 0.95]	0.83	[0.61, 1.04]	.68	[.44, .79]	.83**		
								R2 = .681**	
								95% CI[.44,.79]	
(Intercept)	11.83	[-5.74, 29.41]							
complaints	0.69**	[0.39, 0.99]	0.76	[0.43, 1.08]	.26	[.05, .46]	.83**		
privileges	-0.10	[-0.37, 0.17]	-0.10	[-0.38, 0.17]	.01	[-.03, .04]	.43*		
learning	0.25	[-0.07, 0.56]	0.24	[-0.07, 0.54]	.03	[-.04, .10]	.62**		
raises	-0.03	[-0.40, 0.35]	-0.02	[-0.35, 0.30]	.00	[-.01, .01]	.59**		
								R2 = .715**	Delta R2 =
								95% CI[.42,.79]	95% CI[-.04,

Note. Note. A significant b-weight indicates the beta-weight and semi-partial correlation are also significa
b represents unstandardized regression weights. beta indicates the standardized regression weights. sr2
represents the semi-partial correlation squared. r represents the zero-order correlation. Square brackets ar
used to enclose the lower and upper limits of a confidence interval. * indicates p < .05. ** indicates p <

```
# Format the names
colnames(regrot) <- c("Predictor", "$b$", "$b$ 95\\% CI", "$\\beta$",
                      "$\\beta$ 95\\% CI", "Fit", "Difference")

regrot[ , 1] <- c("(Intercept)", "Complaints", "", "", "", "(Intercept)",
                  "Complaints", "Privileges", "Learning", "Raises",
                    "", "", "")
```

The only thing left is to format the r-squared and delta r-squared notations. But we can't go and fix *just* the letters, because `"R2 = .681**"` is one string. To make a dynamic table, we have to pull the numbers from their source, put the appropriate stars, and merge it with our fancy notation. While I was snooping around the *apaTables* source code, I decided to write the functions for it myself. Check out Table 6 for the result.

```
r_squared_tidy <- function(tableobject, x) {
  #tableobject is the apa.reg.table() object
  #x is the block number you want the r2 from

  R2 <- apa(tableobject$table_block_results[[x]]$model_summary_extended$r.squared, 3, F)
  R2_p <- tableobject$table_block_results[[x]]$model_summary_extended$p.value
  R2_star <- if(R2_p < .01) { "**" }
    else if (.01 <= R2_p & R2_p < .05) { "*" } else ""
  # I wanted the significance stars to be consistent with
  # what apa.reg.table() does, but you can tweak R2_star to add *** for < .001

  fancyR2 <- "$R^2$ = "
```

```
  output <- paste(c(fancyR2, R2, R2_star), collapse = "")
  return(output)
   }

delta_r_squared_tidy <- function(blk2,blk1,tableobject) {
  # blk2 and blk1 are lm() objects for the second and first block,
  # tableobject is the apa.reg.table() object
  R2_2 <- tableobject$table_block_results[[2]]$model_summary_extended$r.squared
  R2_1 <- tableobject$table_block_results[[1]]$model_summary_extended$r.squared

  deltaR2 <- apa(R2_2 - R2_1, 3, F)
  deltaR2_test <- anova(blk2,blk1)
  deltaR2_p <- deltaR2_test$`Pr(>F)`[2]
  sigstar <- if(deltaR2_p < .01) { "**" }
    else if (.01 <= deltaR2_p & deltaR2_p < .05) { "*" } else ""

  # I wanted the significance stars to be consistent with
  # what apa.reg.table() does but you can tweak sigstar to add *** for < .001

  fancydelt <- "$\\Delta R^2$ = "

  output <- paste(c(fancydelt, deltaR2, sigstar), collapse = "")

  return(output)
}

### Keep in mind that both of these functions depend on "MOTE" package
```

```
### because I used apa(), so make sure it's loaded.
### I hope these help! Feel free to modify them as you like!

regrot[3,6] <- r_squared_tidy(regro, 1)

regrot[11,6] <- r_squared_tidy(regro, 2)

regrot[11,7] <- delta_r_squared_tidy(regres, regres0, regro)

### Notice that we can also use LaTeX code in the table note
apa_table(regrot, caption = "Dependent Variable: Overall department ratings",
note = "A significant b-weight indicates the beta-weight is also significant.
$b$ represents unstandardized regression weights.
$\\beta$ indicates the standardized regression weights.
* indicates p < .05. ** indicates p < .01.", font_size = "small", escape = F,
    align= c("l", "r", "c", "r", "c", "l", "l"),
          placement = "p")
```

TIP: If you want to compare multiple models side by side in one table, you can check out the *stargazer* package. It gives you neat results, though not in APA style.

How about correlation matrices? Fortunately, `apa.cor.table()` requires almost zero extra work here, other than optionally capitalizing the variable names. You only need to make sure it fits the page once you have too many variables; landscape format and smaller fonts will help. You could also decide to remove the means and standard deviations and

Table 6

Dependent Variable: Overall department ratings

Predictor	*b*	*b* 95% CI	β	β 95% CI	Fit	Difference
(Intercept)	14.38*	[0.82, 27.94]				
Complaints	0.75**	[0.55, 0.95]	0.83	[0.61, 1.04]		
					$R^2 = .681$**	
					95% CI[.44,.79]	
(Intercept)	11.83	[-5.74, 29.41]				
Complaints	0.69**	[0.39, 0.99]	0.76	[0.43, 1.08]		
Privileges	-0.10	[-0.37, 0.17]	-0.10	[-0.38, 0.17]		
Learning	0.25	[-0.07, 0.56]	0.24	[-0.07, 0.54]		
Raises	-0.03	[-0.40, 0.35]	-0.02	[-0.35, 0.30]		
					$R^2 = .715$**	$\Delta R^2 = .034$
					95% CI[.42,.79]	95% CI[-.04, .11]

Note. A significant b-weight indicates the beta-weight is also significant. *b* represents unstandardized regression weights. β indicates the standardized regression weights. * indicates $p < .05$. ** indicates $p < .01$.

present them elsewhere.

```
matr <- apa.cor.table(attitude, filename = NA)


apa_table(matr$table.body, caption = matr$table.title, note = matr$table.note,
          font_size = "footnotesize", row.names = F,
          placement = "p")
```

TIP: Keep in mind that you're not bound to one way of displaying correlations. Let's say you have "central" constructs and a bunch of other ones (e.g., Big 5 and some behavior measures), but their correlation matrix doesn't fit the page. If you only care about how the central ones are related to other measures, you can just leave the columns for the important ones. Make sure they come first in the data frame so they appear at the top of the rows, and the left of the columns. Then just remove their rows, and the columns that come after theirs; this will make it much easier to fit the page. You can present the means, SDs and intercorrelations for the important variables separately.

Other challenges

There are a few other things you should consider when knitting to .pdf. Even when you set your encoding to utf-8, you might have rendering errors due to some special characters. The error message will give you the Unicode numbers so you might detect and remove them, but sometimes these are invisible characters literally designed to occupy empty space! They usually come from text that you copy and paste into RMarkdown. It's okay to write your paper in your preferred environment (e.g. Word, Docs) and still benefit from *papaja* templates, but be aware of these issues.

Table 7

Means, standard deviations, and correlations with confidence intervals

Variable	M	SD	1	2	3	4	5	6
1. rating	64.63	12.17						
2. complaints	66.60	13.31	.83** [.66, .91]					
3. privileges	53.13	12.24	.43* [.08, .68]	.56** [.25, .76]				
4. learning	56.37	11.74	.62** [.34, .80]	.60** [.30, .79]	.49** [.16, .72]			
5. raises	64.63	10.40	.59** [.29, .78]	.67** [.41, .83]	.45* [.10, .69]	.64** [.36, .81]		
6. critical	74.77	9.89	.16 [-.22, .49]	.19 [-.19, .51]	.15 [-.22, .48]	.12 [-.25, .46]	.38* [.02, .65]	
7. advance	42.93	10.29	.16 [-.22, .49]	.22 [-.15, .54]	.34 [-.02, .63]	.53** [.21, .75]	.57** [.27, .77]	.28 [-.09, .58]

Note. Note. M and SD are used to represent mean and standard deviation, respectively. Values in square brackets indicate the 95% confidence interval. The confidence interval is a plausible range of population correlations that could have caused the sample correlation (Cumming, 2014). * indicates $p < .05$. ** indicates $p < .01$.

The other issue with .pdf is that the format isn't meant to be edited. It may be frustrating to see tiny, annoying mistakes in your knitted document that you easily could've fixed in Word, but don't exactly know how to do with RMarkdown. For instance, some non-English characters may not be properly rendered, LaTeX can put some unnecessary vertical space,[2] or some functions may return extra output that you cannot suppress. .pdf is pretty much inaccessible unless you have some advanced software, but remember that we have another output file which comes before that: .tex! You can open the .tex file in a LaTeX editor such as Overleaf, which works on your browser so you don't have to download anything. It gives you some buttons for basic formatting (bold, italics, etc.) and puts the rendered document next to your code so you can see. You probably won't need much LaTeX knowledge to fix small things, but even if you do, the website has really helpful guides and tutorials. If you're having problems inserting LaTeX code into your table, just paste the `apa_table()` console output in Overleaf's editor; it's much more informative than RMarkdown's error message. And finally, some problems (such as the badly-rendered non-English characters) may be due to RMarkdown's .tex-to-.pdf rendering. When you open the .tex in Overleaf and compile the .pdf from there, some of them will disappear. Give it a shot!

Final remarks

I'm glad you've made it to the end. I hope this guide will make your writing and reporting more efficient and error-free. You may have figured out that I'm far from an expert in R or LaTeX; I probably have oversimplified or overcomplicated some things. These were what I've learned from just tinkering with the code until I got something that works. Hopefully, thanks to this guide, you'll be less likely to resort to trial-and-error. But I also hope that now you won't be afraid to tweak things when necessary, whether in R or

[2] `apa6` LaTeX class offers the `noextraspace` option (like the ones we had in the YAML front matter) to fix this. The manual also says "use at your own risk", so it may not work perfectly.

LaTeX. Thanks for reading, and good luck!

Just so you can see how the references look, let me end by thanking R (Version 3.6.3; R Core Team, 2020).

References

R Core Team. (2020). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/